UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Towards Protection Against Low-Rate Distributed Denial of Service Attacks in Platform-as-a-Service Cloud Services

(Versão Definitiva Após Defesa Pública)

Bruno Carneiro da Silva

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º Ciclo de Estudos)

Orientador: Prof. Doutor Mário Marques Freire

Covilhã, Dezembro de 2018

Dissertation prepared at Instituto de Telecomunicações, within IT Branch – Covilhã, by Bruno Carneiro da Silva, Bachelor of Science in Network Management and Information Technology by Faculdade Maurício de Nassau (Brazil), advised by Dr. Mário Marques Freire, Senior Researcher of the Instituto de Telecomunicações and Full Professor of the Department of Computer Science at Universidade da Beira Interior, and submitted to Universidade da Beira Interior for obtaining the degree of Master of Science in Computer Science and Engineering.

Dedicatory

This work is specially dedicated to two people who have been supporting me through my entire life.

For all the the unconditional and endless love, trust and all the encouragement.

To my father and my mother.

Acknowledgements

Since I have started to work with technology, I have been in constant learning with different people, from different countries and cultures. Whether having conversations about personal matters or share knowledge about professional and academic development. I would like to express my gratitude to all of them.

To my advisor, Professor Doctor Mário Marques Freire, for the availability, acceptance, guidance and tutoring throughout this work.

To all the teachers who have given me a piece of their minds in order for me to achieve my goals and increase my thirst for knowledge.

To everyone who are very dear to me and were extremely important in this journey, for their support and remarkable memories. A Special heartwarming thank you to girlfriend, Tamires Ferreira Nascimento.

To my mother, Iara Carneiro da Silva, who believes in me and encourage every decision I have been making in my life.

To my father, Jorge Gomes da Silva, who taught me a lot and helped my mother raising me to become the person that I am.

Abstract

Nowadays, the variety of technology to perform daily tasks is abundant and different business and people benefit from this diversity. The more technology evolves, more useful it gets and in contrast, they also become target for malicious users. Cloud Computing is one of the technologies that is being adopted by different companies worldwide throughout the years. Its popularity is essentially due to its characteristics and the way it delivers its services. This Cloud expansion also means that malicious users may try to exploit it, as the research studies presented throughout this work revealed. According to these studies, Denial of Service attack is a type of threat that is always trying to take advantage of Cloud Computing Services.

Several companies moved or are moving their services to hosted environments provided by Cloud Service Providers and are using several applications based on those services. The literature on the subject, bring to attention that because of this Cloud adoption expansion, the use of applications increased. Therefore, DoS threats are aiming the Application Layer more and additionally, advanced variations are being used such as Low-Rate Distributed Denial of Service attacks. Some researches are being conducted specifically for the detection and mitigation of this kind of threat and the significant problem found within this DDoS variant, is the difficulty to differentiate malicious traffic from legitimate user traffic. The main goal of this attack is to exploit the communication aspect of the HTTP protocol, sending legitimate traffic with small changes to fill the requests of a server slowly, resulting in almost stopping the access of real users to the server resources during the attack.

This kind of attack usually has a small time window duration but in order to be more efficient, it is used within infected computers creating a network of attackers, transforming into a Distributed attack. For this work, the idea to battle Low-Rate Distributed Denial of Service attacks, is to integrate different technologies inside an Hybrid Application where the main goal is to identify and separate malicious traffic from legitimate traffic. First, a study is done to observe the behavior of each type of Low-Rate attack in order to gather specific information related to their characteristics when the attack is executing in real-time. Then, using the Tshark filters, the collection of those packet information is done. The next step is to develop combinations of specific information obtained from the packet filtering and compare them. Finally, each packet is analyzed based on these combinations patterns. A log file is created to store the data gathered after the Entropy calculation in a friendly format.

In order to test the efficiency of the application, a Cloud virtual infrastructure was built using OpenNebula Sandbox and Apache Web Server. Two tests were done against the infrastructure, the first test had the objective to verify the effectiveness of the tool proportionally against the Cloud environment created. Based on the results of this test, a second test was proposed to demonstrate how the Hybrid Application works against the attacks performed. The conclusion of the tests presented how the types of Slow-Rate DDoS can be disruptive and also exhibited promising results of the Hybrid Application performance against Low-Rate Distributed Denial of Service attacks. The Hybrid Application was successful in identify each type of Low-Rate DDoS, separate the traffic and generate few false positives in the process. The results are displayed in the form of parameters and graphs.

Keywords

Denial of Service, Security, Cloud Computing, Packet Analysis, Application Layer

Resumo

Actualmente, a variedade de tecnologias que realizam tarefas diárias é abundante e diferentes empresas e pessoas se beneficiam desta diversidade. Quanto mais a tecnologia evolui, mais usual se torna, em contraposição, essas empresas acabam por se tornar alvo de actividades maliciosas. *Computação na Nuvem* é uma das tecnologias que vem sendo adoptada por empresas de diferentes segmentos ao redor do mundo durante anos. Sua popularidade se deve principalmente devido as suas características e a maneira com o qual entrega seus serviços ao cliente. Esta expansão da *Computação na Nuvem* também implica que usuários maliciosos podem tentar explorá-la, como revela estudos de pesquisas apresentados ao longo deste trabalho. De acordo também com estes estudos, *Ataques de Negação de Serviço* são um tipo de ameaça que sempre estão a tentar tirar vantagens dos serviços de *Computação na Nuvem*.

Várias empresas moveram ou estão a mover seus serviços para ambientes hospedados fornecidos por provedores de *Computação na Nuvem* e estão a utilizar várias aplicações baseadas nestes serviços. A literatura existente sobre este tema chama atenção sobre o fato de que, por conta desta expansão na adopção à serviços na *Nuvem*, o uso de aplicações aumentou. Portanto, ameaças de *Negação de Serviço* estão visando mais a camada de aplicação e também, variações de ataques mais avançados estão sendo utilizadas como *Negação de Serviço Distribuída de Baixa Taxa*. Algumas pesquisas estão a ser feitas relacionadas especificamente para a detecção e mitigação deste tipo de ameaça e o maior problema encontrado nesta variante é diferenciar tráfego malicioso de tráfego legítimo. O objectivo principal desta ameaça é explorar a maneira como o protocolo HTTP trabalha, enviando tráfego legítimo com pequenas modificações para preencher as solicitações feitas a um servidor lentamente, tornando quase impossível para usuários legítimos aceder os recursos do servidor durante o ataque.

Este tipo de ataque geralmente tem uma janela de tempo curta mas para obter melhor eficiência, o ataque é propagado utilizando computadores infectados, criando uma rede de ataque, transformando-se em um ataque distribuído. Para este trabalho, a ideia para combater *Ataques de Negação de Serviço Distribuída de Baixa Taxa* é integrar diferentes tecnologias dentro de uma Aplicação Híbrida com o objectivo principal de identificar e separar tráfego malicioso de tráfego legítimo. Primeiro, um estudo é feito para observar o comportamento de cada tipo de Ataque de Baixa Taxa, a fim de recolher informações específicas relacionadas às suas características quando o ataque é executado em tempo-real. Então, usando os filtros do programa Tshark, a obtenção destas informações é feita. O próximo passo é criar combinações das informações específicas obtidas dos pacotes e compará-las. Então finalmente, cada pacote é analisado baseado nos padrões de combinações feitos. Um arquivo de registo é criado ao fim para armazenar os dados recolhidos após o cálculo da Entropia em um formato amigável.

A fim de testar a eficiência da Aplicação Híbrida, uma infra-estrutura *Cloud* virtual foi construída usando OpenNebula Sandbox e servidores Apache. Dois testes foram feitos contra a infra-estrutura, o primeiro teste teve o objectivo de verificar a efectividade da ferramenta proporcionalmente contra o ambiente de Nuvem criado. Baseado nos resultados deste teste, um segundo teste foi proposto para verificar o funcionamento da Aplicação Híbrida contra os ataques realizados. A conclusão dos testes mostrou como os tipos de *Ataques de Negação de Serviço Distribuída de Baixa Taxa* podem ser disruptivos e também revelou resultados promis-

sores relacionados ao desempenho da Aplicação Híbrida contra esta ameaça. A Aplicação Híbrida obteve sucesso ao identificar cada tipo de *Ataque de Negação de Serviço Distribuída de Baixa Taxa*, em separar o tráfego e gerou poucos falsos positivos durante o processo. Os resultados são exibidos em forma de parâmetros e grafos.

Resumo Alargado

A evolução tecnológica está sempre acelerada e seus benefícios como disponibilizar mais automação, agilidade e facilidade às actividades diárias, são os motivos da constante adopção de tecnologias no uso pessoal e nos negócios. A cada ano, várias novas tecnologias surgem, outras são actualizadas e utilizadas de uma forma diferente, sendo empregadas em áreas distintas. Novas técnicas e práticas tecnológicas estão sempre sendo adoptadas e melhoradas, não seria diferente quando se trata de *Computação na nuvem*.

A *Computação na Nuvem* não é uma ideia completamente nova mas sua ampla adopção recente é notável. Uma tecnologia que beneficia várias outras áreas como medicina, economia, turismo, academia e mesmo para uso pessoal, *Computação na Nuvem* se tornou uma das tecnologias mais consolidadas da actualidade. Por isso, também se tornou uma tecnologia muito visada por usuários maliciosos. Como *Computação na Nuvem* trabalha com a noção dos recursos fornecidos serem virtuais e ilimitados, vários *Hackers* já se aproveitam dessa gama de recursos disponibilizados pelas tecnologias voltadas à Nuvem para realizar actividades suspeitas e criminosas.

Uma das tecnologias que mais se beneficiam da *Computação na Nuvem* são aplicações para dispositivos móveis. Estas aplicações utilizam a Nuvem para suprir as mais diversas áreas e efectuar diferentes tarefas como: Comércio, sistemas empresariais, jogos, navegação, armazenamento de informações como palavras-passe, ficheiros, correio electrónico e fotos. Para efectuar tantas tarefas e não perder o foco em suas características, *Computação na Nuvem* utiliza de Data Centers geolocalizados em diferentes parte do mundo, o que faz essa tecnologia possuir bastante poder computacional. Por esse motivo, também acaba se torna um paraíso para a prática de ataques massivos na Internet como *Ataques de Negação de Serviço Distribuídos*.

Pesquisas mostradas ao longo desse trabalho, explicam brevemente as características das principais tecnologias que habilitam o uso da *Computação na Nuvem*, como: O início do uso de Mainframes Corporativos e Sistemas Compartilhados por Tempo; A utilização da Virtualização como forma de usufruir melhor dos recursos dos servidores; Arquitectura Orientada à Serviço para unir de uma maneira mais satisfatoria práticas empresariais com a tecnologia; A adopção de Serviços Web, o qual utilizam da capacidade da Internet para substituir a Intranet e assim suprir necessidades de comunicação e interacção com os clientes. O objectivo é entender como os ataques à *Computação na Nuvem* podem ser na verdade específicos contra essas tecnologias que juntas possibilitam os serviços na Nuvem.

Existem vários tipos de ameaças e mais ameaças continuam surgindo a cada momento. Novos métodos de segurança também são criados acompanhando a evolução tecnológica e como resultado, as ameaças também se encontram em constante evolução. Os desafios de proteger uma tecnologia em ascensão como *Computação na Nuvem* são grandes, especialmente quando se trata de uma tecnologia que é habilitada por várias outras. Problemas técnicos, organizacionais ou de gestão relacionados aos Mainframes, à Virtualização, à Arquitectura Orientada à Serviços e aos Serviços Web também são herdados pela *Computação na Nuvem*.

A disponibilidade de serviços na Nuvem são divididos em Modelos de Serviços e de Distribuição. Cada modelo tem suas características que podem ser estudadas para melhor suprir a necessi-

dade de cada cliente. Esses modelos são acedidos de forma online, o que deixa os serviços relacionados à *Computação na nuvem* dependentes da Internet. Por consequência, de acordo com estudos abordados ao longo deste trabalho, *Ataques de Negação de Serviço* específicos na camada de aplicação estão em crescimento.

Ataques de Negação de Serviço são ataques que abusam da quantidade de tráfego enviados a um servidor ou determinado serviço para deixa-lo inoperante e assim parar de permitir o acesso de cliente legítimos durante o período do ataque. Esse tipo de ameaça possui muitas variantes podendo ser utilizado em diferentes camadas dos Modelos de Comunicação. Esse ataque ainda pode ser feito em conjunto, utilizando vários computadores que efectuam o ataque em simultâneo, transformando-o em um ataque distribuído.

Diversos são os motivos que levam usuários maliciosos a praticar ataques de negação de serviços: Motivos activistas ou políticos, demonstrações criminosas de poder, motivos de protesto ou até mesmo sabotagem profissional. Esses ataques são feitos em diferentes áreas de negócios que utilizam tecnologias informáticas como: Em sites de jogos online ou relacionados à apostas; Directamente à empresas com intuito de extorquir ou prejudicar os serviços da mesma os deixando inacessíveis; Websites governamentais ou relacionados à algum evento específico. Os ataques distribuídos podem ser feitos quando um atacante (*Master*) utiliza de computadores que foram previamente infectados e agora fazem parte de uma rede de Ataque de Negação de Serviço. Há também dentro desta rede de ataques distribuídos, pessoas que simplesmente se voluntariam para usar seu computador pessoal e fazer parte do ataque, expandindo ainda mais o nível do *Ataque de Negação de Serviço* e o dano feito ao alvo especificado.

Existem diferentes variantes dos *Ataques de Negação de Serviço*. As variantes abordadas neste trabalho são ICMP e UDP flood, ataque TCP SYN, inundação HTTP e técnicas de Amplificação e Reflexão de ataques. Essas variantes têm em comum o fato de serem usadas como um ataque volumétrico, o qual a principal característica é preencher as requisições feitas ao servidor alvo abruptamente utilizando de inúmeras conexões, e assim deixar o servidor inacessível para clientes legítimos. Vários métodos de defesa existem para mitigar esse tipo de ataque, porém os ataques de negação de serviço também continuam evoluindo. Outras variantes do Ataque de Negação de Serviço são mais sofisticadas, conseguindo explorar aplicações ou usar os métodos de comunicação de protocolos de segurança para alcançar o objectivo final de deixar o serviço inoperante. Estas variantes também conseguem mesclar-se à tráfegos legítimos utilizando tráfego em baixa taxa.

Os *Ataques de Negação de Serviço de Baixa Taxa (LDoS)*, não utilizam a forma abrupta de enviar numerosas quantidades de pacotes para deixar serviços indisponíveis. Esta variante utiliza a forma como os protocolos interagem de acordo com a arquitectura cliente-servidor, principalmente o protocolo HTTP, para enviar tráfegos em baixa taxa como se fossem pacotes enviados de clientes legítimos. Porém são tráfegos maliciosos o qual ocupam conexões a fim de negar o acesso de clientes legítimos à determinados serviços. Essa variante também pode ser usada como ataque distribuído da mesma forma que os *Ataques de Negação de Serviços* mais tradicionais, efectuando o ataque sincronizadamente com múltiplos computadores, sejam eles controlados remotamente ou espontaneamente voluntários.

Os métodos de defesas tradicionais existentes contra *Ataques de Negação de Serviço*, não são

suficientes para detectar ou mesmo mitigar ataque considerados mais avançados como LDoS. Empresas especializadas se dedicam a tentar amenizar os danos desse tipo de ataque e a identificar o mesmo o mais rápido possível. Porém, algumas destas soluções podem ser caras ou simplesmente requerer um nível de customização muito alto para certos clientes. Na literatura existem poucas pesquisas dedicadas exclusivamente aos tipos de *Ataque de Negação de Serviço Distribuída de Baixa Taxa*. Uma análise sobre estas pesquisas é feita neste trabalho e algumas conclusões são atribuídas para servirem como base literária referidas em um único local. Além disso, as pesquisas também servem para entender os diferentes métodos criados e testados contra *Ataques de Negação de Serviço Distribuída de Baixa Taxa*, para desenvolver um método diferente para mitigar este tipo de ameaça.

Neste trabalho, o objectivo principal é criar um mecanismo de protecção para a camada de aplicação contra *Ataques de Negação de Serviço Distribuída de Baixa Taxa*. As pesquisas citadas ao longo do trabalho, foram utilizadas como base para desenvolver uma solução que utiliza um método baseado em múltiplos testes e observação dos ataques em tempo real. A observação destes padrões é necessária para combater directamente os pacotes que possuem características que podem resultar em um ataque e também assim criar uma solução leve, simples e eficiente contra LDDoS na camada de aplicação.

Para isso, foi desenvolvida uma Aplicação Híbrida em linguagem C que utiliza o programa Tshark para capturar os pacotes e a partir daí, extrair informações necessárias para a próxima fase da aplicação. As informações extraídas são: O endereço IP da origem, o endereço IP de destino, tamanho da janela de anúncio, tamanho total do pacote recebido, código de status do protocolo HTTP e tamanho do conteúdo. Estas informações são referentes às características dos tipos de *Ataque de Negação de Serviço Distribuída de Baixa Taxa*. Após a colecta destas informações, combinações são feitas baseando-se na quantidade que um endereço IP aparece associado igualmente ao valor de uma das outras informações extraídas. Assim, a Aplicação Híbrida faz comparações destes padrões e caso o pacote seja considerado um ataque, é descartado. Caso não seja considerado um ataque, o pacote segue. Um arquivo de registo é criado com informações em um formato para melhor leitura e análise. Falso-positivos são analisados para futuras melhorias no método proposto.

Para efectuar os testes, um ambiente *Cloud* virtual foi criado a partir de dois computadores em *Cluster* para usufruir de características como balanceamento de carga e alta disponibilidade. Dentro do ambiente de produção foram instalados dois servidores Web com Apache. O objectivo do primeiro teste foi verificar quais configurações da ferramenta se adequa contra o ambiente criado a ponto de deixá-lo inoperante. Desta forma, com os resultados obtidos, pode-se utilizar estas configurações como base para efectuar o segundo teste.

Após o ajuste da ferramenta de ataque, o segundo teste é feito directamente contra o serviço HTTP dos servidores Apache que agora possuem a Aplicação híbrida desenvolvida como protecção contra os *Ataques de Negação de Serviço Distribuídos de Baixa Taxa*. Deste modo, pôde-se analisar a eficiência da Aplicação Híbrida e obter os resultados dos ataques feitos ao *Cluster*. A Aplicação Híbrida apresentou bons resultados em identificar e mitigar o *Ataque de Negação de Serviço Distribuída de Baixa Taxa*, apenas gerando poucos falsos positivos. Portanto foi concluído que o método obteve sucesso e gerou bons resultados gerais contra os *Ataques de Negação de Serviço Distribuída de Baixa Taxa* na camada de aplicação.

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
app	Applications
CGI	Computer Graphical
CPU	Central Processing Unit
CRLF	Carriage Return Line Feed
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DX-DoS	Distributed XML Denial of Service
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IPv6	Internet Protocol version 6
ISA	Instruction Set Architecture
KVM	Kernel-based Virtual Machine
LDDoS	Low-Rate Distributed Denial of Service
LDoS	Low-Rate Denial of Service
NIST	Network Institute of Technology
OWASP	Open Web Application Security Project
QoS	Quality of Service
REST	Representational State Transfer
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SOA	Service-Oriented Architecture
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TTL	Time to Live
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
VLSI	Very Large Scale Integration
VM	Virtual Machine
VOIP	Voice Over Internet Protocol
VPN	Virtual Private Network
WAF	Web Application Firewall
WSDL	Web Services Description Language
XXE	XML External Entity

Chapter 1

Introduction

1.1 Focus and Scope

The rapid evolution of technology brings different services and various option to accelerate the daily work of millions of people in distinct fields. For that reason, as revealed in the literature [6, 7, 8], researches related to technology are frequently supported. Several findings from these researches incorporate new methods to use and adapt distinct technologies within numerous other areas. This scenario has led many Chief Information Offices to revitalize their legacy core systems because many traditional systems that run office processes had become hamstrung by accumulated technical debt and dependencies [9].

Cloud computing is attractive to business owners because eliminates the requirement for users to plan ahead for provisioning, and allows enterprises to start small and increase resources only when there is a increase in service demand [10]. Since Cloud Computing has useful features, its services may be used in several ways and companies from different fields are relying even more on Cloud to move their business, as illustrated in figure 1.1. As a consequence, attacks are focusing on it. Cloud Services and datacenters became the target of countless threats in the past years.

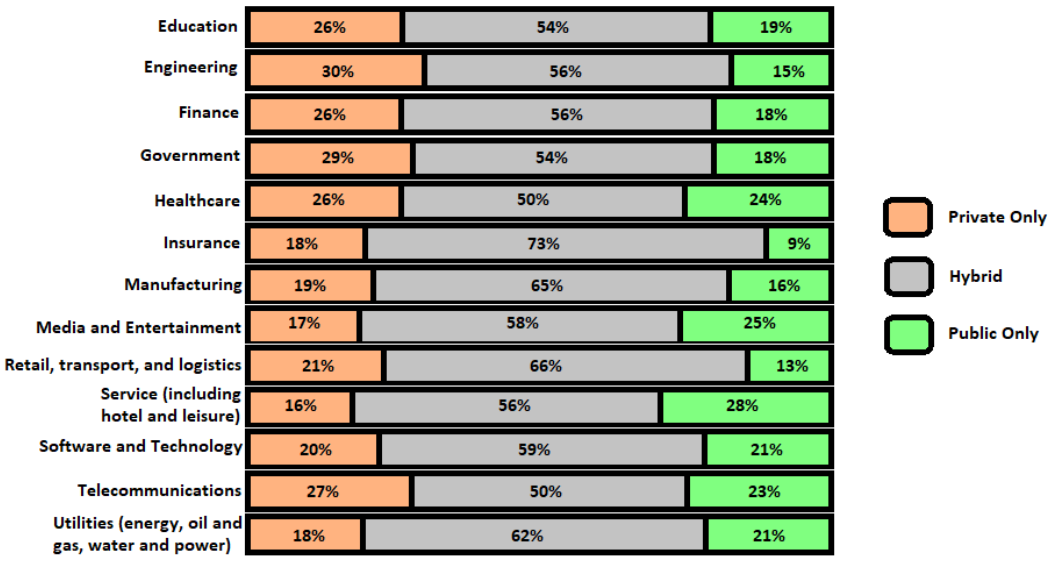

Figure 1.1: Cloud Models adoption throughout different business (Adapted from [1])

A special type of attack that makes security professionals on high alert is the Denial of Service attack (DoS). This threat, as the name suggests, is a malicious attempt by a single person or a group of people to make the target deny services to its customers. When this attempt derives from a single host of the network, it constitutes a Denial of Service attack. On the other hand, it is also possible that several malicious hosts coordinate to flood the target with an abundance

of malicious packets, so that the attack takes place simultaneously from multiple points. This type of attack is called a Distributed Denial of Service or DDoS attack [11].

Denial of Service attack is dangerous because it has different variants in which some of them, makes this threat even more powerful and hard to detect. This is the case of Low-Rate Denial of Service (LDoS), which one of its characteristics is not send high traffic volumes as a classic DoS attack. Instead, the attacker sends or receives packets slowly in order to make a service or a server unavailable by upholding the sockets open until the communication finishes. This type of attack differs from other kinds of DoS variants, because it aims the Application Layer and sends legitimate packets abusing the way HTTP Protocol works.

There are several methods which detects classic Denial of Service threats but not as much methods that detects the Low-Rate Denial of Service variant. The access to Cloud Computing Services is done using the Internet, usually with an installed platform on a computer, an app installed on a mobile device or a Web Browser. With this, the number of applications also grew, and alongside with it, more threats aiming the Application Layer emerged.

The focus of this work is to develop a method to protect Cloud Computing services against Low-Rate Distributed Denial of Service and test in different ways. In order to achieve that, a crucial study about the Modus Operandi of Cloud Computing and its enabled technologies is done. Then, an explanation about Denial of Service attacks and its variants, focusing in the Low-Rate Distributed Denial of Service. In order to test the application, a dataset from the Western Regional Collegiate Cyber Defense Competition [12] is used to test the application against different traffic. A testbed environment will be created consisting in a virtualized simulated environment that receives real-time attacks.

1.2 Problem Statement and Objectives

Attacks can be considered a collection of activities which aims to compromise confidentiality, integrity and availability, which are the pillars of systems built to secure data, software, hardware or network resources. Attacks known as Denial of Service can stop companies services. According to the Akamai Security reports [13], throughout the years, there is an increase in the amount of traffic that DoS attacks are using to hit different businesses. In September of 2016, the IT host french based company OVH was affected by a DoS type of attack which achieved the peaks of over 1Tb per second. They estimate that the attack came from about 152.000 devices, from Closed Circuit Television cameras to Smartphones [14, 15]. In October of the same year, the servers of Dyn, a company that controls the majority of the DNS infrastructure on the Internet, was hit by a series of DDoS attacks which reports disclosed that the attack carried out 1.2Tb per second peaks, and remained under sustained assault for most of the day, bringing down sites including Twitter, the Guardian, Netflix, Reddit, CNN and other sites in Europe and the United States [16, 17]. These are examples of how powerful Denial of Service attacks are. These attacks are made using infected devices to create a large network and then, launch the attack when they are commanded to.

A study released by the company Imperva Incapsula [18] - which is a company that delivers Web Application Firewalls in the Cloud - revealed in their quarterly Global DDoS Threat Report

that Application Layer attacks continue to increase. According to their study, more than half of all attacks in the last quarter of 2017, were between 100-1,000 rates per second, which is an increase from 43.5 percent compared to the third quarter of 2017. Their data points to an increase in activity by non-professional offenders, who typically mount smaller sized assaults using attack scripts and DDoS-for-hire services.

This means that different models of Distributed Denial of Service attacks are being used and the Low-Rate Denial of Service attack is one of them. Because of its characteristics of sending slow and legitimate packets, it can blend into normal traffic. This attack was spotted being the favorite method of attack used by hacktivists groups as seen in the Iranian Elections in 2009 [19]. Another typical use of this feature is to masquerade other attacks. While servers struggle with the LDDoS attack, hackers can use this moment to gain privileged access, infect the machine or even install a Backdoor.

The problem addressed in this work is the fact that Low-Rate Denial of Service attacks is hard to detect due to its nature and existing methods on the market can be expensive or difficult to assemble.

Based on the perspectives acquired about Cloud Computing usability and the problem evaluated, the main objective of this work is to develop a lightweight method that differentiate malicious traffic from legitimate traffic achieving the minimum numbers of false positives possible. As a result, specifically protect against Low-Rate Distributed Denial of Service attacks. Also develop a testbed to test the capability and efficiency of the method created. The secondary objectives to be achieved are:

1. Present a single research work that joins important technologies behind Cloud Computing Infrastructure and its security features. As well as the relation between Cloud and Denial of Services Attack.
2. Provide an understanding about the features of Denial of Service attacks. Explaining details of its variants and present the evolution from classic flood types to more sophisticated ones. Emphasizing on Low-Rate Distributed Denial of Service.
3. Provide analysis of researches and different approaches in order to learn how distinct methods operate to defend against *Low-Rate DDoS* attacks.

1.3 Approach for Solving the Problem

Throughout this work, several defense mechanisms with different methods will be explained. Other technologies that try to mitigate DDoS threats exists but there are elements missing in these legacy security systems to battle more refined attacks such as Low Rate Distributed Denial of Service. There are some researches using other approaches such as mathematical methods, flow counting and Information Theory. Each one of those approaches with their own particular vision of the problem.

The main goal of this work is try to separate malicious traffic from legitimate traffic, reducing the false positive. Therefore, there is a necessity of an accurate method to detect when a LDDoS attack is occurring. In order to achieve that, this work presents an Hybrid Application

that combines Packet analysis and combination methods after a pattern analysis of the LDDoS attack in rea-time. Therefore, a more exact way to separate malicious traffic from legitimate traffic can be done before it reaches the target services and applications. The use of already existing approaches such as Apache modules were chosen as a base inspiration because they already have some type of mechanism defense that can be used against LDDoS, but they are not that effective when working alone as Pascoal demonstrated in [20].
The Hybrid Application uses Tshark to capture and separate the instructions used by the different types of Low-Rate Distributed Denial of Service attacks. Then, these instructions are combined with the objective of search for matches between the information inside the packets and specific characteristics of the LDDoS types. The application also creates a log file to store particular information about the IP address considered suspicious and its respective packets. This way, further analysis can be done about these incoming traffic.

1.4 Contributions

Other researches such as Ain [21] and Xiang [22] already proved that Information Theory methods are efficient against Distributed Denial of Service Attacks. Therefore, the main contribution of this work is a different method which incorporates already existent traditional technologies specifically to fight the Low-Rate Distributed Denial of Service variant, tested on a suitable testbed environment in real-time and over a newer dataset for the test of the Hybrid Application capability.

Other contribution that should be mentioned is the use of a newer dataset for the test of the Hybrid Application capability and the creation of a testbed environment.

1.5 Dissertation Overview

This dissertation is arranged within 5 chapters and appendixes which possess the codes used to built the Hybrid Application. The remaining of the dissertation is organized as follow:

The chapter I is the introduction of the dissertation, which establishes the focus and scope, the specification of the problem statement and explanation of the objectives. Also, the approach to solve the problem is discussed and clarified as well as the contributions and the total organization of the dissertation.

The Chapter II is focused on the state of the art study of Cloud Computing and Low-Rate Distributed Denial of Service attacks. The background of Cloud Computing is presented with the focal point being the security issues alongside Denial of Service attacks, focusing on Low Rate Distributed Denial of Service. An explanation about the Cloud Computing general operation, Cloud Services, enabled technologies and Security challenges are pointed. Also, a study about Distributed Denial of Service attacks is presented and the state of art of Low Rate Distributed Denial of Service attacks is detailed. Its concepts, types, operation methods, existing protection against this threat and the current challenges in detecting and mitigating it. Lastly, the related research on different approaches to fight against Low Rate Distributed Denial of Service attacks.

In chapter III, the Testbed implementation is described. The simulation environment, technologies used including the dataset chose and the proposed architecture of the Hybrid Application are detailed in this chapter. The tests and results are also disclosed. Details about the test implementations such as the explanation of the commands used to perform the attacks, the effectiveness of the attack tool and the evaluation of the Hybrid Application. The efficiency in mitigating Low Rate Distributed Denial of Service attacks are justified.

In chapter IV, the main conclusion is presented based on the development of the whole work and directions for future work are provided.

Chapter 2

State of the Art Study

2.1 Introduction

The term Cloud Computing was coined in 1997 by Ramnath Chelappa, an Information Systems professor at the University of Texas, but the idea of Cloud Computing dates back from the end of 1950's and early 1960's, when Mainframes started to be used in business to process data and Professor John McCarthy [23] suggested that computer time-sharing technology might lead to a future where computing power and even specific applications might be sold through a utility-type business model. Numerous researchers and engineers from other technology related areas such as Software Engineering, Database, Network, Server Administration, Developers and several other distinct fields are connecting with Cloud Computing.

In this Chapter, a State of the Art study about Cloud Computing is presented. In Section 2.2, Cloud Computing Background stages are discussed, from its main characteristics to its Deployment and Service Models, as well as briefly described aspects of Cloud Enabled Technologies, its Security concerns and the connection between Denial of Service attacks and Cloud Computing.

Among various Internet based attacks, Denial of Service (DoS) attack is a critical and continuous threat in cyber security. In general, DoS attacks are implemented by either forcing a victim computer to reset, or consuming its resources, e.g., CPU cycles, memory or network bandwidth. As a result, the targeted computer can no longer provide its intended services to its legitimate users [24]. When this threat is performed using more computers in an hierarchical stance, this attack is considered a distributed attack.

Also in this chapter, Distributed Denial of Service Attacks are analyzed. In section 2.3, details about the concepts, characteristics and motives behind the execution of these attacks. As well as an explanations about DDoS architecture, and how this threat is gradually implemented. DDoS different types, from general to advanced are interpreted and defense techniques that challenge these types are described. In section 2.4, the focus transition to a specific advanced DDoS type called Low-Rate Denial of Service attack and its capabilities to target the Application-Layer. Its particular types and operation methods are explained, including a study about traditional defense techniques against Low-Rate DDoS and the challenges in detection this attack.

Finally, in section 2.5, Related researches on Low-Rate DDoS detection methods are analyzed and discussed in the end of this chapter in section 2.6.

2.2 Cloud Computing Background

Accordingly to Sosinsky [25], Cloud Computing refers to applications and services that run on a distributed network using virtual resources and accessed by common Internet protocols and

networking standards. It is distinguished by the notion that resources are virtual and limitless and details of the physical systems on which the software runs are abstracted from the user.

There are several definitions which try to explains Cloud Computing. For this work, the Network Institute of Standards and Technology (NIST) definition, characteristics, service and deployment models are used. However, other sources are incorporated with the NIST ones, with the purpose to create a wider understandable approach of this technology and its capabilities.

The NIST[26], defines Cloud Computing as being a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. The Cloud model that NIST suggests, is composed of five essential characteristics, three service models, and four deployment models. These can be considered the base of Cloud Computing which are used nowadays. However, other literature such as the Cloud Security Alliance, Borko Furht & Armando Escalante, Zhang et. al and Goldszmidt & Poddar, respectively [27, 28, 10, 29], Suggest there other crucial features may be included as part of Cloud characteristics. Also, Juan-Verdejo and Surajbali, Deloitte Consulting Team and Duan et. al, respectively in [30, 9, 31], stated that the Cloud Service Models can also be expanded.

2.2.1 Cloud Main Characteristics

For this work, nine characteristics from the literature aforementioned are considered:

- **On-Demand Self Service**: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider [26].
- **Broad Network Access**: Before, resources were available only inside mainframes or, later though Intranet. Now, with Cloud Computing they can be available from anywhere considering the place where the access is being made has an Internet connectivity available. These services hosted in the Cloud are generally web-based. Therefore, they can be easily accessible through a variety of devices with Internet connections. These devices not only include Desktop and Laptop computers, but also Smartphones and Personal Digital Assistents [10].
- **Resource Pooling**: Cloud Computing works though Data Centers. Physical and logical resources of a Cloud Provider are merged together and distributed between numerous costumers, rather be private users or departments within companies. The Cloud is also an autonomous system and it is managed transparently to users. Hardware, Software and data inside the Cloud can be automatically reconfigured, orchestrated and consolidated to present a single platform image, finally rendered to users [32].
- **Rapid Elasticity**: The scalability and flexibility are important features of Cloud Computing. Cloud Services could be scaled across various concerns, such as geographical locations, hardware performance or software configurations. The computing platforms should be flexible to adapt to different requirements of a potentially large number of users. [32].

- **Measured Service**: Cloud systems automatically control and optimize resource usage by leveraging a metering capability at some level of abstraction appropriate to the type of service, for example, Storage, processing, bandwidth, or active user accounts. Resource usage can be monitored, controlled, and reported — providing transparency for both the provider and consumer of the service [27].
- **Multi-Tenancy**: The ability to deliver software to multiple client Organizations - or tenants - from a single, shared instance of the Software is an important requirement for Web-delivered solutions[29]. The resources inside the Cloud infrastructure are shared between these different type of clients, but the costumers do not have contact with the resources of one another.
- **Geo-Distribution**: In order to achieve high network performance and localization, several Cloud Providers possess Data Centers located at different locations around the globe. A service provider can easily leverage geo-diversity to achieve maximum service utility [10].
- **Resiliency**: Because of the Geo-Distribution and rapid elasticity characteristics, combined with Virtualization technology, maximum uptime and reliability is an important factor when using Cloud Computing. The capabilities for fast recovery, in case of a disaster or simply a mistake, are provided on account of replications of data which are previously created. In case of a failure, backup instances of the application, platform or infrastructure service must be ready to replace the failed services without disruption [33].
- **Quality of Service (QoS) and Pricing**: Cloud computing does not require up-front investment. No capital expenditure is required. Users pay for services and capacity as they need them. Also, Cloud Computing can guarantee QoS in terms of hardware/CPU, such as performance, bandwidth, and memory capacity for users [28]

Considering these characteristics, safety and reliability are common factors that have a important role in Cloud Computing. To be able to provide large numbers of resources to distinctive costumers and maintain them available anytime, anywhere, it requires several security features against numerous threats.

Creating a wider explanation about Cloud Computing and its characteristics is part of the relevance to understand the reason this technology is being adopted in distinct business areas.

2.2.2 Cloud Deployment Models

Deployment Models refers to the location and management of the Cloud infrastructure. These Models are Cloud types: Private, Public, Community and Hybrid Clouds. Service Models consist of the particular types of services that can be access on a Cloud Computing platform [25].

2.2.2.1 Deployment Models

According to the NIST [26], The Deployment Models are:

- **Private Cloud**: The Private Cloud infrastructure is operated for the exclusive use of an Organization. The Cloud may be managed by that Organization or a mediator. Private Clouds may be either on-premises or off-premises. Based on a Cloud Computing Infrastructure, it

is assumed that a Private Cloud does not employ the same level of Virtualization or pooling of resources that a Cloud Computing Provider can achieve [25].

- **Public Cloud**: A Public Cloud is built over the Internet and can be accessed by any user who has paid for the service [34]. This model is available to the general public and resources are shared among consumers. Public Clouds offer several key benefits to service providers, including no initial capital investment on infrastructure and the shifting of risks to infrastructure providers [10].

- **Community Cloud**: A Community Cloud is organized to serve a common function or purpose. It may be for one Organization or for several Organizations that share common concerns such as their mission, policies, security or regulatory compliance needs. A Community Cloud, equally to a Private Cloud, may also be managed by the constituent Organization(s) or by a mediator [25].

- **Hybrid Cloud**: Hybrid Cloud is a combination of two or more Cloud Models - Private, Community or Public Cloud - where those Clouds retain their unique identities, but are bound together as a unit. A Hybrid Cloud may offer standardized or proprietary access to data and applications, as well as application portability [25]. In a Hybrid Model, companies can maintain production applications on-site while conducting all their testing in the cloud. This enables on-demand scaling of test environments as needed and eliminates the cost of underutilized hardware [35].

In summary, Public Clouds promote standardization, preserve capital investment, and offer application flexibility. Private Clouds attempt to achieve customization and offer higher efficiency, resiliency, security, and privacy. Hybrid Clouds operate in the middle, with many compromises in terms of resource sharing [34].

2.2.3 Cloud Service Models

There are three predominant services within Cloud Computing according to the NIST [26], *Software as a Service*, *Platform as a Service* and *Infrastructure as a Service*. These Services, their respective structure and the control aspects for the clients and providers are displayed in (2.1).

- **Infrastructure as a Service (IaaS)**: Infrastructure as a Service (IaaS) is the delivery of hardware (Servers, Storage and Network) and associated software (Virtualization technology, File Systems), as a Service. The IaaS provider does limited management besides than keep the Data Center operational [36]. It allows costumers to have most of the control over their environment. The processing, memory and other resources are monitored, and depending on the Service Level Agreement made with the provider, they can be monitored by the provider, or the costumer itself.

  Public Infrastructure as a Service providers commonly offer virtual servers containing one or more CPUs, running several choices of Operating Systems and a customized software

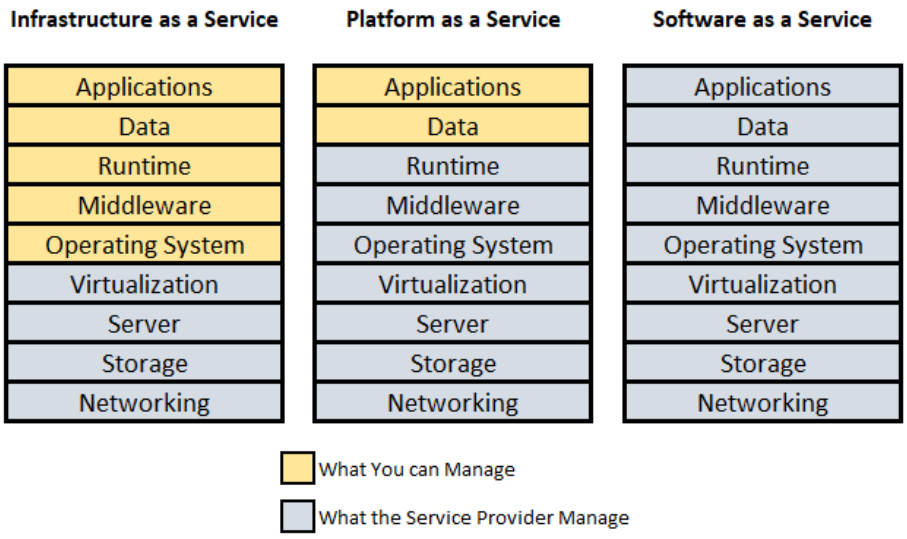

Figure 2.1: Cloud Service Model (Adapted from [2])

stack [37]. As costumers are provided with the infrastructure, they have control over the type of Operating System, the Database, Middleware and Applications they choose to deploy and install.

Within this service model, other components may be included in the Infrastructure, depending on the choice of Cloud Vendor and its service availability, such as:

- Desktop Virtualization
- High-Availability
- Backups
- Load Balancing
- Hardware Security

- **Platform as a service (PaaS)**: This Service Model gives the client the possibility to deploy its applications on the Cloud Infrastructure or use applications that were programmed using languages and tools that are supported by the PaaS service provider [25]. The users also have control over the data and the application. In addition, in some cases, even control over the hosting environment configuration.

  PaaS supplies all the resources required to build applications and services completely from the Internet, without having to download or install any software. PaaS services include application design, development, testing, deployment, and hosting. Other services include team collaboration, web service integration, database integration, security, scalability, storage, state management, and versioning. [35].

  Based on that, one may conclude that the security of the application inside the Platform is responsibility of the clients, as the providers work to maintain a safe Infrastructure underlying the Platform.

- **Software as a Service (SaaS)**: Is a software delivery model, which provides customers access to business functionality remotely (usually over the Internet) as a service. The customer does not specifically purchase a software license. The cost of the infrastructure, the right to use the software, and all hosting, maintenance and support services are all bundled into a single monthly or per-use charging [38].

  On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are kept rather low, compared with conventional hosting of user applications. Customer data is stored in the Cloud that is either a Cloud Vendor proprietary or publicly hosted to support PaaS and IaaS [34].

  Basically, the costumer is given access to systems such as a Costumer Relationship Software, a Financial Management System, an Enterprise Resource Planning, an Accounting or Collaboration systems, which are not hosted inside a company but is hosted remotely at the Cloud Data Center of a Cloud Provider.

2.2.4 Cloud Computing Enabled Technologies

Cloud Computing may be considered a technology which relies on the benefits and characteristics of other technologies to provide its services. As explained in [34], computing technology has undergone a series of platform and environment changes in the past 60 years. The distributed computing became data-intensive and network-centric and as a result, a necessity emerged to upgrade data centers using fast servers, storage systems, and high-bandwidth networks.

Throughout the literature [25, 37, 34, 28] several technologies are mentioned as being Cloud enabling technologies. Although specifically for this work, Mainframes and Time-share systems, Service-Oriented Architecture, Web Services and Virtualization are the ones to be considered. The main reason is to establish a connection between Cloud Computing Enabled Technologies and Denial of Service attacks. This threat is recognized as been an active threat against these technologies, as demonstrated in [27, 39, 40, 41]. Another reason to choose these specific technologies among others in the literature is to understand Cloud Computing characteristics, Deployment and Service Models. Because the manner in which Cloud delivers its services may be directly connected with these technologies and their evolutionary process.

2.2.4.1 Mainframes, Clusters and Time Sharing Systems

Utility Computing is the on demand delivery of infrastructure, applications, and business processes in a security-rich, shared, scaleable, and standards-based computer environment over the Internet for a fee [37]. Several Technologies have in some way aimed at turning the Utility Computing concept into reality. Companies which offered common data processing tasks, such as payroll automation, operated time-shared mainframes as utilities, which could serve several applications and often operated close to 100% of their capacity. [37]

A Mainframe is the central data repository, or hub, in a corporation Data Processing Center, connected to users through less powerful devices such as workstations or terminals. The presence of a Mainframe often implies a centralized form of computing, as opposed to a distributed

form of computing [42].

During the 1950s and 1960s, Mainframes were devoted almost entirely to the processing of computationally intensive tasks demand for computers, as data processing machines expanded and new machines were built to meet this demand [43], including for example, massive calculations, of which assisted the man landing on the moon for the first time [7]. Part of the mainframe popularity and longevity may be associated with its inherent reliability and stability, a result of careful and steady technological advances that have been made since the introduction of the System/360 [44].

Time-sharing was an idea that emerged in 1961. over the next decade or two, many time-sharing systems were developed [45].The maturity of the mainframe models during the late 1960s and 1970s, as presented by Jim Elliot from IBM at [43], demonstrated that Virtual Storage Capabilities were added to the System/360 model and later integrated in the 370 model. It was the First IBM system with virtual storage capabilities and also, Time Sharing System was the official Operating System from the IBM Data System Division.

Time Sharing System has more then one interpretation according with Corbató et. al. [46], one can mean using different parts of the hardware at the same time for different tasks, or several persons making use of the computer at the same time. The first meaning, often called multiprogramming, is oriented towards hardware efficiency in the sense of attempting to attain complete utilization of all components. The second Meaning, is primarily concerned with the efficiency of persons trying to use a computer.

In mainframe computing, dummy terminals acted as user interface devices, but stand-alone Personal Computers became powerful enough to meet the majority of the necessity of these users, providing local computing power and cashing support [28]. Before Personal Computers and intelligent workstations became popular [44], from 1960s to 1980s, lower-cost mini-computers became popular among small businesses and on college campuses. From 1970 to 1990, it happened an extensive use of PCs built with VLSI microprocessors [34].

Also, in the early 1990s, the client/server model of computing with its distributed nodes of less powerful computers, emerged to challenge the dominance of Mainframe computers [44]. Client/Server model is a convenient and natural extension of inter-process communication on a single machine and its interaction forms the basis of most network communication [47], such as Internet Services (web, mail, ftp, streaming, etc), telecommunication systems (IMS, VoIP, IPtv, etc.) and enterprise applications (information services, databases, etc.) [28].

A enterprise might have a large server collection that includes transaction servers, database servers, email servers, and web servers. Large collections of servers are sometimes called server farms [42]. Cloud-based services require large computing capacity and are hosted in data centers and server farms. These distributed data centers and server farms span multiple locations and can be linked via inter-networks providing distributed computing and service delivery capabilities [48].

But similar to Mainframes, computing servers and desktop computers in a modern organization are often underutilized [37]. According with the SUSE Project [49], Virtualization brings a lot

of advantages while providing the same service as a hardware server. Although Virtualization it is not a new science, its use begin to develop in early 2000 [50, 51]. Also, at the same time, Abstraction and a move toward computing as a public utility took a great step forward when the concept of Software as a Service (SaaS) was developed [52].

Servers are mainly used to provide service to a customer, and a virtualized Operating System can provide the same service [49] while addressing some problems related to Mainframes and Servers such as:

- **Underutilized Servers**: Virtualization allows organizations to partition a single physical computer or server into several virtual machines [53]. This may imply that server machines acquired by companies are usually robust in resources, but the company only uses a portion of those resources. Therefore, the server becomes underutilized.
- **Physical Space and Scalability**: Data Center could be a large room in the basement of a building or a room full of servers in any part of the world that can be accessed via Internet [35]. Hence, one can presume that this requires space and also, physical servers are limited to the hardware it possesses. Mainframes scale vertically by adding more computing, storage or connectivity resources [28]. By creating multiple resources from a single computer or server, Virtualization improves scalability and workloads [53], since Virtual Machines can be created and removed according to necessity.
- **Power Limitations**: Distributed servers result in power and cooling requirements per square foot that stress current Data Center power thresholds [42]. One may consider that Mainframes which are maintained inside Organizations will consume a large amount of power. Therefore, in order to hold these mainframes, Organizations must comply with the costs of electricity or even have the costs of change that, in some cases, cloud be necessary to keep large consuming machines. Virtualization results in the use of fewer overall servers, less energy consumption, and less infrastructure costs and maintenance [53].
- **Availability**: Depending on the server usage, it needs to be available all the time. For internal or external use. The company must have a procedure to monitor the availability of its servers.
- **Specialized IT Support Team**: Mainframes require fewer staff when supporting hundreds of applications. Because centralized computing is a major theme when using the mainframe, many of the configuration and support tasks are implemented by writing rules or creating a policy that manages the infrastructure automatically [42]. Since Virtualization let a physical machine be divided into several Virtual Machines, one can assume that using this technology may require even less IT support Team, specially for the Hardware Infrastructure.

There are fewer Mainframes in use today than there were 20 years ago because of corporate mergers and data center consolidations. In some cases, applications were moved to other types of systems. The consolidation effect has produced powerful mainframes, which might need only 1% of their power to run an older application [42]. As presented by Rich Uhlig at. al [51], Full Virtualization of all system resources – including processors, memory, and I/O devices – makes it possible to run multiple Operating Systems on a single physical platform.

Today, as stated by Mike Ebbers et. al [42] even if there is a decrease in the mainframes utilization, they still have a central role in the business world. Considering its evolution from in-house generated computing power into utility-supplied computing resources delivered over the Internet as Web services [37].

Cloud Computing is, in many ways, a return to the centrally coordinated integration of the mainframe time-share era, being a counter reformation that brings the possibility of real performance gains for users and their organizations [52].

This evolutionary process, according with authors [23, 28, 54] composes technologies that may represent crucial elements and foundation technologies for Cloud Computing. Cloud resources are virtualized and service-oriented. That is, everything is expressed and exposed as a service [37]. Some focal points in the development of these base technologies may be expressed:

- **Virtualization**: The launch of the CP/CMS Architecture, by the IBM Cambridge Scientific Center, which were a group of compatible time-sharing system with the first implementation of a mainframe operating system with Virtualization support. [42].
- **Service-Oriented Architecture**: According with Thomas Erl [55], after repeated generations of traditional distributed solutions, the severity of problems such as Integration, inefficiency and resource waste, resulting in complex infrastructures and convoluted Enterprise Architecture, has been amplified. Those are the reason Service-Orientation was conceived. It represents an evolutionary state in the history of IT in that it combines successful design elements of past approaches with new design elements that leverage conceptual and technology innovation. A Service-Oriented Architecture also includes a set of cloud services, which are available on various distributed platforms [28].
- **Web Services**: Web Service initial use was primarily within traditional distributed solutions wherein they were most commonly used to facilitate point-to-point integration channels. As the maturity and adoption of Web services standards increased, so did the scope of their utilization [55].

  Web Services can incorporate applications running on different messaging product platforms, enabling information from one application to be made available to others, and enabling internal applications to be made available over the Internet [37]. Cloud services are typically designed as Web services, which follow industry standards including WSDL, SOAP, and UDDI [28].

As disclosed in this section, one can cite the evolution of Cluster Servers and Virtualization Technology as factors for the reduction in Mainframes use. As well as the maturity of Time Sharing Systems from simple terminals to Personal Computers and from PCs to Client/Server distributed systems. As a result, Cloud Computing may be considered enabled by these technology progress.

2.2.4.2 Service-Oriented Architecture

Enterprise software has always suffered from the mismatch between technical and business-related concepts and the different languages spoken by the people on both sides. As a result, the development not only faced inefficiencies, but also suffered the lost of important knowledge and consequently, many solutions had to be reinvented. It is a key goal of an Service-Oriented

Architecture to provide services that have a concrete meaning on the business level [54].

A Service-Oriented Architecture (SOA) is one in which IT supports the business processes that cover current and emerging requirements to run the business end-to-end. SOA unifies business processes by structuring large applications as a collection of smaller modules known as *Services*. SOA presents a design framework for realizing rapid and low-cost system development and improving total system quality [35]. Figure 2.2 demonstrates a simple Service-Oriented Architecture Framework.

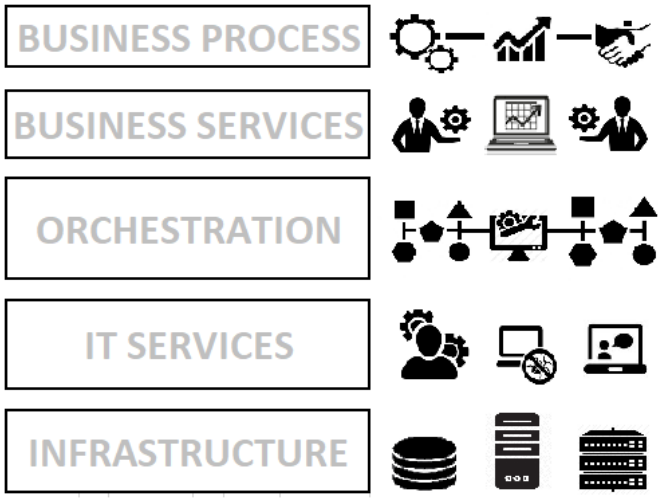

Figure 2.2: Service-Oriented Architecture Framework

Virtual resources and computing assets are accessed through the Cloud, including not only externally hosted services but also those provided globally by companies. Therefore, placing information, services, and processes outside the enterprise without a clear strategy is not productive. An SOA solution consists of a linked set of business services that realize an end-to-end business process. At a high level, SOA can be viewed as enabling improved management control, visibility, and metrics for business processes. [23].

According to Krafzig et. al. [54] Service-Oriented Architecture is based on the key concepts of an Application front-end, Service, Service Repository, and Service Bus. They are briefly explained below:

- **Front-End Programming**: Initiate and control all activity of the enterprise systems. There are different types of application front-ends. An application front-end with a graphical user interface, such as a Web application or a rich client that interacts directly with end users can be cited as examples. However, application front-ends do not necessarily have to interact directly with end users. Batch programs or long-living processes that invoke functionality periodically are also valid examples of application front-ends.

  It is possible that an application front-end delegates much of its responsibility for a business process to one or more services. Ultimately, however, it is always an application front-end that initiates a business process and receives the results.

- **Services**: The term *Service* is not generic. It has specific connotations that relate to a unique combination of design characteristics. When solution logic is consistently built

as services and when services are consistently designed with these common characteristics, service-orientation is successfully realized throughout an environment [55]. A service is a software component of distinctive functional meaning that typically encapsulates a high-level business concept. It consists in several parts: A contract, interfaces, implementation, business logic and data.

- **Enterprise Service Bus**: A service bus connect all the participants of an SOA - services and application front-ends - with each other. If two participants need to communicate - for example, if an application front-end needs to invoke some functionality of a basic service - the service bus provides this function. The service bus is not necessarily composed of a single technology, but rather comprises a variety of product and concepts. Its main characteristics are: Connectivity, Heterogeneity of technology and of communication concepts and technical services.

- **Service Repository**: A Service Repository provides facilities to discover services and acquire all information to use the services. Although much of the required information is already part of the service contract, the service repository can provide additional information such as physical location, information about the provider, contact persons, usage fees, technical constraints, security issues, and available service levels.

Different design paradigms exist for distributed solution logic. What distinguishes service-orientation is the manner in which it carries out the separation of concerns and how it shapes the individual units of solution logic. For that, eight principles of designing have to be met, in order to build an application using Service-Oriented Architecture. These principles, as Thomas Erl stated in [55], are:

- **Service Contract:** Services express their purpose and capabilities via a service contract. A great deal of emphasis is placed on specific aspects of contract design, including the manner in which services express functionality, how data types and data models are defined, and how policies are asserted and attached. There is a constant focus on ensuring that service contracts are both optimized, appropriately granular, and standardized to ensure that the endpoints established by services are consistent, reliable, and governable.

- **Service Loose Coupling:** The principle of Service Loose Coupling promotes the independent design and evolution of a service's logic and implementation while still guaranteeing baseline interoperability with consumers that have come to rely on the service's capabilities. There are numerous types of coupling involved in the design of a service, each of which can impact the content and granularity of its contract.

- **Service Abstraction:** Abstraction ties into many aspects of service-orientation. On a fundamental level, this principle emphasizes the need to hide as much of the underlying details of a service as possible. Doing so directly enables and preserves the previously described loosely coupled relationship. Service Abstraction also plays a significant role in the positioning and design of service compositions.

- **Service Re-usability:** Reuse is strongly advocated within service-orientation; so eminently, that it becomes a core part of typical service analysis and design processes, and also forms the basis for key service models. The advent of mature, non-proprietary service technology has provided the opportunity to maximize the reuse potential of multi-purpose logic on an unprecedented level.

- **Service Autonomy:** For services to carry out their capabilities consistently and reliably, their underlying solution logic needs to have a significant degree of control over its environment and resources. The principle of Service Autonomy supports the extent to which other design principles can be effectively realized in real world production environments. Isolation levels and service normalization considerations are taken into account to achieve a suitable measure of autonomy, especially for reusable services that are frequently shared.

- **Service Statelessnes:** The management of excessive state information can compromise the availability of a service and undermine its scalability potential. Services are therefore ideally designed to remain stateful only when required.

- **Service Discoverabiility:** For services to be positioned as IT assets with repeatable Return of Investment they need to be easily identified and understood when opportunities for reuse present themselves. The service design therefore needs to take the "communications quality" of the service and its individual capabilities into account, regardless of whether a discovery mechanism (such as a service registry) is an immediate part of the environment.

- **Service Composability:** The ability to effectively compose services is a critical requirement for achieving some of the most fundamental goals of service-oriented computing. Complex service compositions place demands on service design that need to be anticipated to avoid massive retro-fitting efforts. Services are expected to be capable of participating as effective composition members, regardless of whether they need to be immediately enlisted in a composition.

The implementation of SOA can be observed in recent studies disclosed by Soares et al [56], where an analysis of specific design principles for SOA implementation is done. It shows how SOA is being used in practice in different activities such as refactoring legacy systems, integration of systems with the purpose of producing a new system, and to create new methodologies to develop systems. Also in [25], Sosinsky states that PaaS offers the benefits of cloud computing and is often composed and based on a Service-Oriented Architecture model. Other Service-Orientation Architecture implementation in real world are demonstrated in [55, 54], where case studies are implemented within companies. The case studies displayed suitable projects for the companies exclusive necessities, as well as analysis of the technical infrastructure, definition of project scopes, benefits, business goals, obstacles and perspective.

As reported by Feuerlicht in [57], The transition from the traditional vertically integrated business and IT structures towards SOA presents a number of important challenges. These challenges range from purely technical issues such as performance of the SOAP protocol and the maturity Web Services standards, to business issues that include considerations of skills availability and SOA infrastructure costs. For this work, two challenges related to these business issues and technical issues, may be considered when connecting SOA features with Cloud Computing and threats such as Denial of Service Attacks, Governance and Security. They are briefly described below:

- **Governance**: SOA Governance is concerned with establishing policies, controls and enforcement mechanism that contributes to the success of implementing Service-Oriented Architecture. The Governance will mark the difference between the success and failure of SOA and has a lot to do with Service-Oriented Architecture management as it determines

the roles of IT professionals, standards within the organization, ownership, allocation of resources and project delivery lifecycles [58]. //

- **Security**: SOA is built on open standards such as XML, WSDL, SOAP, UDDI and these standards do not have security of their own. This makes services and systems vulnerable to attacks [58]. While the core Web Services standards (i.e. XML, SOAP, WSDL, and UDDI) are relatively mature and stable, many of the additional standards that address important issues such as security and reliability (e.g. WS-Coordination, WS-Atomic Transaction, WSDM, WS-Reliability, etc.) are still under development [57].

The services organization and orchestration inside Cloud could be managed in a Service-Oriented Architecture (SOA). A set of Cloud Services furthermore could be used in a SOA application environment, thus making them available on various distributed platforms and could be further accessed across the Internet [32].

Service-Oriented Architecture and Cloud Computing are close to each other. Specifically mentioning, SOA is an architectural pattern that guides business solutions to create, organize and reuse its computing components, while Cloud Computing is a set of enabling technology that services a bigger, more flexible platform for enterprise to even build their SOA solutions. In other words, SOA and cloud computing will co-exist, complement, and support each other [59].

2.2.4.3 Web Services

Web Services technology is the most promising choice to implement Service-Oriented Architecture and its strategic objectives. A Web service is essentially a semantically well-defined abstraction of a set of computational or physical activities involving a number of resources, intended to fulfill a customer need or a business requirement. A Web service could be described, advertised and discovered using standard-based languages, and interacted through Internet-based protocols [60].

The shift towards SOA has been facilitated by the unprecedented level of standardization based around Web Services core standards: XML, SOAP, WSDL, and UDDI, that address service security, reliability, transactions, coordination and management. The universal acceptance of the core Web Services standards by the industry produced a situation where for the first time it is technically possible for applications to interact across diverse computing environments without incurring massive integration costs [57].

A Web Service is a software system designed to support inter-operable machine to machine interaction over a network. It has an interface described in a machine process able format (specifically Web Services Description Language). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web related standards [61]. The figure 2.3 displays the basic functionality of a Web Service.

Web Services move beyond helping various types of applications to exchanging information. The technology also plays an increasingly important role in accessing, programming on, and integrat-

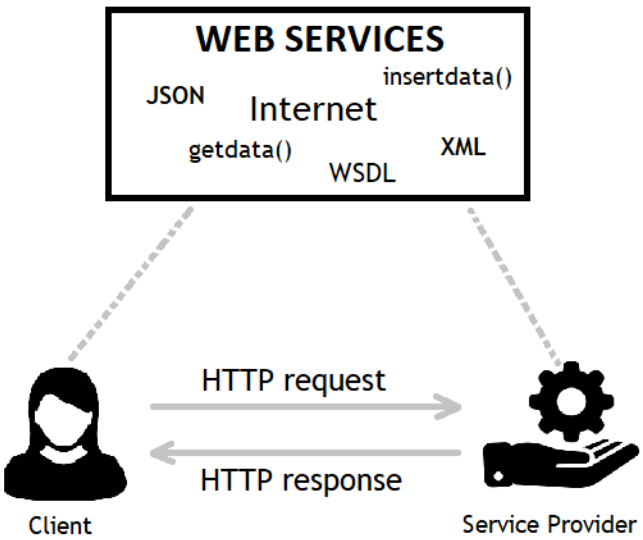

Figure 2.3: Basic Web Services

ing a set of new and existing applications [34].

Services are based on three XML-based technologies, which have been adopted as standards for interoperability [58]. These standards are developed by organizations such as the W3C (World Wide Web Consortium) [62, 63] and OASIS (Organization for the Advancement of Structured Information Standards) [64] to establish a common format for developers. These standards are:

- **Simple Object Access Protocol(SOAP)**: It is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics [62].

- **Web Service Description Language (WSDL)**: It Provides a model and an XML format for describing Web services. WSDL 2.0 enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as "how" and "where" that functionality is offered. This specification defines a language for describing the abstract functionality of a service as well as a framework for describing the concrete details of a service description. It also defines the conformance criteria for documents in this language [63].

- **Universal Description, Discovery and Integration (UDDI)**: It is a framework for doing what it suggests: describing, discovering, and integrating (business) services via the Internet. The UDDI framework uses SOAP to communicate with programs that access it [65].

Any web service that complies with the basic specification (SOAP, WSDL and UDDI), can be considered a SOAP-based Web Service. Exchange of messages happens over a default format, XML. The XML messages have an already defined SOAP structure, which must contain and envelope, an header and a body. Detailed information can be found in [62]

In order to send requests to a Web Provider, REST is another approach besides SOAP messages used to access Web Services.

Representation State Transfer(REST) is an architectural approach which emphasize on using HTTP methods and URI conventions for simpler and lightweight communication on the web. A web service implemented using principles of REST is called a RESTFul service [66].

It can be used to create any service. Unlike SOAP, there is no central body that defines a strict specification method in a RESTful service, but must follow certain principles when designing a type of service for communication between two or more applications. Those principles are: Uniform interface, Stateless, Cacheable, Layered System, Code on demand. More detailed information about them can be seen in [67].

Since Web Services uses Open standards such as XML, WSDL, UDDI and SOAP, Service-Oriented Architecture can be thoroughly implemented using this technology. Although, Web services are not the only way to implement SOA, but has been identified as the best technology that fully implements SOA potentials [58]. Web Service can be labeled as the one which uses SOA principles and elements as a whole. Simply put, Web services are about technology specifications, whereas SOA is a software design principle. Notably, Web services Description Language is an SOA-suitable interface definition standard, and this is where Web services and SOA fundamentally connect [68].

Cloud Computing services are normally exposed as Web services, which follow the industry standards such as WSDL , SOAP and UDDI 25 [32]. Web Service delivered from the Cloud may include:

- Vendors outsource Infrastructure-as-a-Service, it relies heavily on modern on-demand computing technology and high-speed networking [23]. This model deliver its resources over remote connections, VPN and FTP in case of VMs and storage services [69].
- Some vendors who provide Software-as-a-Service (SaaS), foray into the implementation issues, the characteristics, benefits, and architectural maturity level of the service [23]. In case of web applications, its resources may be accessed through web browsers using HTTP/HTTPS protocols [69].
- Outsourced hardware environments (called platforms) are available as Platforms-as-a-Service [23]. SOAP, REST and RPC Protocols, in case of Web Services and APIs, is how their resources can be accessed [69].

With the expansion of information interchange and sharing arising from the proliferation of the Internet, the value and scope of application of Web services are increasing. As such, Web servers have become the key targets of DDoS attacks, which can inflict the most serious damage in terms of availability [70].

2.2.4.4 Virtualization

Virtualization has revolutionized the Data Center technology through a set of techniques and tools that facilitate the providing and management of the dynamic data center infrastructure. It has become an essential and enabling technology of Cloud Computing environments [37]. As presented by the literature [50, 51, 6, 53, 49], Virtualization itself is an extensive subject. For

this work, the idea is to briefly explain this technology operation, capabilities, benefits, and its role in the importance to Cloud Computing.

Virtualization can be defined as follow:

- Virtualization is a computer architecture technology by which multiple Virtual Machines (VMs) are multiplexed in the same hardware machine. Hardware resources or software resources can be virtualized in various functional layers [34].
- Virtualization creates a simulated, or virtual, computing environment as opposed to a physical environment. Virtualization often includes computer-generated versions of hardware, operating systems, storage devices, and more. This allows Organizations to partition a single physical computer or server into several virtual machines [53].

Basically, one can see that these definitions have some points in common, such as, the fact that Virtualization is a technique which allow physical computer environments (may be a single computer or multiple computers), to operate as multiple Virtual Machines at the same time. This is accomplished via transforming physical hardware into virtual hardware resources. The figure 2.4 displays how a basic Virtualization system works.

It is also important to understand the meaning of Virtual Machine, which is essentially, a virtual computer, that is a logical representation of a computer in a software version [6]. The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility [34]. One can state that a Virtual Machine is an isolated software reproduction of the original physical computer. They are, in theory, completely independent from one another. Inside a physical computer, it may be possible to run several distinctive guest operational systems, each one of them, running its own processor instruction and resource access.

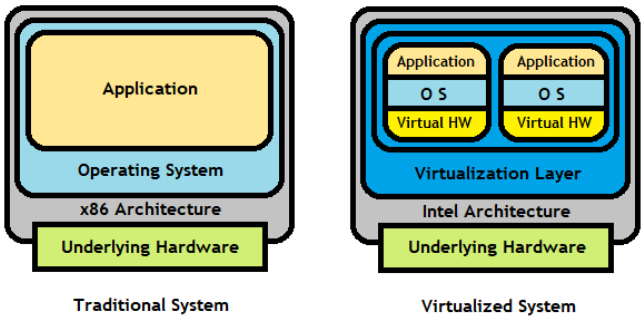

Figure 2.4: Basic Virtualization System (Adapted from [3])

The main function of the software layer for Virtualization is to abstract the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. This can be implemented at various operational levels [34], some of which will be briefly addressed below:

- **Instruction Set Architecture (ISA) Level**: Virtualization is performed by emulating a given ISA by the ISA of the host machine. The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one. One source instruction may require tens or hundreds of native target instructions to perform its function [34].

- **Hardware Abstraction Level**: Hardware-level Virtualization is performed right on top of the bare hardware. This approach generates a virtual hardware environment for a VM and the process manages the underlying hardware through Virtualization. The idea is to virtualize computer resources, such as its processors, memory, and I/O devices and the intention is to upgrade the hardware utilization rate by multiple users concurrently [34].

- **Operating System Level**:This kind of Virtualization partitions the physical machines resources, creating multiple isolated user-space instances, also called Containers. While hypervisor-based Virtualization provides abstraction for full guest OS's (one per virtual machine), container-based Virtualization works at the operation system level, providing abstractions directly for the guest processes [71].

- **User-Application Level**: This is the isolation of applications in the Host machine without modifying this same Host machine, The Operating System, File System, or Registry. With the application Virtualization technology, companies can easily deploy custom or commercial applications across the Organization without installation conflicts and system changes. Some advantages of this type of operational level Virtualization can be that it ensures fast spread of the software, full portability, efficiency in application deployment and the virtualized applications does not require major modifications or security permissions for installation [72].

A low-level program is required to provide system resource access to virtual machines, and this program is referred to as the Hypervisor or Virtual Machine Monitor (VMM) [25]. One can add that the Hypervisor creates an illusion of a physical environment. Its function is to create the abstraction layer, called Virtualization layer, to make it possible to multiple VMs, with different Operating Systems, send requests in order to access the same hardware (ie.: Hard Drive or Network Card) and does not initiate conflict problems. There are two types of Hypervisor according to Sosinsky in [25]: Type 1 Hypervisor and type 2 Hypervisor. Also called Native or Bare Metal Hypervisor and Hosted Hypervisor respectively, as explained below:

- **Type 1 Hypervisor**: Also called Native or Bare-Metal Hypervisor, these are software that run directly above the hardware of the host machine. It also monitors the operating system that runs directly above the Hypervisor and also monitors the operating system that runs on the guest machine. This is because the guest machine Operating System runs on a different or isolated level that is directly above the Hypervisor [73].

- **Type 2 Hypervisor**: Also called Hosted Hypervisor because the Hypervisor is hosted or installed on an already existing Operating System and it houses other Operating Systems above it. In this type of Hypervisor any problem occurring with the host Operating System will affect the guest machine Operating System that is running on the Hypervisor and also will affect the Hypervisor itself [72].

Virtualization technologies partition hardware and thus provide flexible and scalable computing platforms. Virtual Machine techniques, such as Xen [50], offers virtualized IT-infrastructures on demand. Virtual network advances, such as VPN, support users with a customized network environment to access Cloud resources [32].

Virtualization is a conversion process that translates unique IT hardware into emulated and standardized software-based copies. Through hardware independence, virtual servers can easily be

moved to another Virtualization host,automatically resolving multiple hardware-software incompatibility issues [74]. Hardware Virtualization may be classified into three main categories: Full Virtualization, partial Virtualization and Para-Virtualization. For each type, the Hypervisor works translating requests in slightly different ways as explained below:

- **Full-Virtualization**: Uses Bare-Metal Hypervisor, that is responsible for loading the guest Operating Systems. Each guest Operating System will run as if they are operating directly on the underlying hardware and will directly interact with the physical memory and disk space. This implementation allows the Hypervisor to isolate the activities of one guest Operating System from the other [75]. According to Velte et. al [35], Full Virtualization has been successful for several purposes such as: Sharing a computer system among multiple users; Isolating users from each other and from the control program and Emulating hardware on another machine.
- **Partial Virtualization**: This technique usually involves a combination of privilege restrictions by user accounts and a virtual File System. The Virtualization found here, typically refers to providing the target application the appearance of its own virtual File System. The difference between Partial Virtualization techniques from Full Virtualization is that Partial Virtualization approaches share the same OS kernel as the host [76].
- **Para-Virtualization**: In this type of Virtualization, a Para-virtualization supporting Hypervisor is installed on the host Operating System which runs over the underlying hardware. This Hypervisor will act as a Virtualization layer, as the host on which the guest operating systems are loaded. Guest operating systems will make the necessary system calls to Hypervisors for the utilization of hardware resource [75]. Para-Virtualization, is better suited in these sorts of deployments, according to Velte et. al [35]: Disaster recovery, Migration and Capacity management.

One can evaluate that the main difference between them, may be the performance. Since the Full Virtualization runs directly on the hardware, its efficiency may be better then Host-based Virtualization types. As a result, greater overhead may occur, caused by the Operation System that is situated between the physical layer and the Virtualization layer. Although, Full virtualization is used by Cloud providers on physical servers to deploy Cloud Services, the others can also be used depending on the necessity or demand of each client.

As presented in [34], Data Center automation means that large volumes of hardware, software, and database resources in these Data Centers can be allocated dynamically to millions of Internet users simultaneously, with guaranteed QoS and cost-effectiveness. This automation process is triggered by the growth of Virtualization products and Cloud Computing services. Hence, one can conclude that Virtualization allowed companies, no matter the size, to increase the manner of which their business is conducted.

For example, most of the problems revealed by mainframes in 2.2.4.1, Hardware Virtualization technology may be able to fix them. Combining the benefits of Virtualization provided by Rich Uhlig et. al from Intel and the SUSE project, respectively [51, 49], some points may be highlighted as important improvements Virtualization Technology brought to nowadays business:

- **Saves Money and energy**: Less hardware means there is no need to invest in electric power, backup power, and cooling if one needs more service [49]. Therefore, there may be also a improved utilization of hardware resources.

- **Saves Time and Space:** By encapsulating a guest state within a VM, Virtualization makes it possible to decouple the guest from the hardware on which it is currently running and to migrate it to a different platform [51]. Since One can run several Operating System on one host, so all hardware maintenance is reduced [49]. With less physical servers, one can assume that better organization of the Virtual Machines is achieved. This may be considered crucial to increase the resolution of problems and the deployment of new VMs can be accomplished faster.

- **Reduced Management:** Using a VM Guest simplifies the administration of an infrastructure [49]. Hence, the necessity of an IT team will depend on the size of the business. Virtualization mitigates Management of upgrades by allowing systems to run legacy and new operating systems concurrently. In addition to facilitating hardware maintenance operations, VM migration can be triggered automatically by workload balancing or failure-prediction agents. Also, Embedding certain system-management functions within a VM can improve client manageability. [51].

- **Minimize risks:** Virtualization provides migration capabilities, live migration and snapshots. These features reduce downtime, and bring an simpler manner to move services from one place to another without any service interruption [49]. Therefore, backup servers, monitoring and load balancing may be also included as services provided by Virtualization technology. With the combination of these elements, the risks may be better managed.

Data center virtualization is an evolutionary process that was started several years ago within mainframe computer rooms, and it has dramatically intensified in the last few years. Its proposed freedom from physical boundaries has produced benefits in each technology area, and much more importantly, from an architectural perspective [77].

Virtualization and cloud computing are fueling growth and making Data Center solutions more efficient and scalable. More specifically, organizations using Virtualization technologies are seeing greater returns and more viability to deal with the growing demands of the economy [77].

2.2.5 Cloud Security Concerns and Denial of Services against Cloud Technologies

Cloud computing systems have been made possible through the use of large-scale clusters, Service-Oriented Architecture (SOA), Web services, and virtualization [37]. As the studies presented throughout this chapter, the idea of Cloud Computing is not new or revolutionary and one may state that much of the Cloud advantages comes from its own characteristics provided by its enabled technologies aforementioned. But due to the rapid maturity of its methods and strategies to facilitate the goals of business agility, Cloud adoption is increasing potentially fast [74].

Regarding security, with the Cloud model you lose control over physical security. In a Public Cloud for example, you are sharing computing resources with other companies. In a shared pool outside the enterprise, you do not have any knowledge or control of where the resources run [23]. Public Clouds lack fine-grained control over data, network and security settings, which

hampers their effectiveness in many business scenarios [10].

Cloud-based services may also result in many mobile IT users accessing business data and services without traversing the corporate network. This will increase the need for enterprises to place security controls between mobile users and cloud-based services. Placing large amounts of sensitive data in a globally accessible Cloud leaves organizations open to large distributed threats—attackers no longer have to come onto the premises to steal data, and they can find it all in the one virtual location [23]. Since Service Providers typically do not have access to the physical security system of data centers, they must rely on the infrastructure provider to achieve full data security [10]. The administrative access is through the Internet rather than the controlled and restricted direct or on-premises connection that is adhered to in the traditional data center model. This increases risk and exposure and will require stringent monitoring for changes in system control and access control restriction [23].

Cloud Security issues may also be inherited from its enabled technologies problems. For example, as aforementioned in 2.2.4.2, one of the challenges for a successful implementation of a Service-Oriented Architecture, is the governance aspect. In terms of Security, since the Governance is concerned with establishing policies, controls and enforcement mechanism [58], One of the most important actions for a security team is to develop a formal charter for the security organization and program. The charter should be aligned with the strategic plan of the organization or company the security team works for. Also, must clearly define the roles and responsibilities of the security team and other groups involved in performing information security functions. Lack of a formalized strategy can lead to an unsustainable operating model and security level as it evolves [23].

Regarding Virtualization, its efficiencies in the Cloud require Virtual Machines from multiple organizations to be co-located on the same physical resources. Although traditional data center security still applies in the Cloud environment, physical segregation and hardware-based security cannot protect against attacks between Virtual Machines on the same server [23].

From a more technical perspective, Virtual Machine abstraction may be considered a strong security feature when using Virtual servers, because of the Hypervisor which is a complex software. As a result, a study from Perez-Botero et al. [78], revealed that researchers have been tackling security concerns on traditional Hypervisors, further motivated by numerous problem reports disclosed for popular Hypervisors (e.g. Xen, KVM, OpenVZ) in a variety of software vulnerability database. Other concern presented in this study, is that this vulnerabilities can enable exploitation from anywhere and can lead to full control over the virtual environment.

Other important technical issues comes from Web Services security problems, which may allow attackers to take advantage of a flawed system inside a Cloud Environment by delegating attacks to specific parts of the environment where these flawed system are running. The latest OWASP top 10 threats report [79] revealed some information about Web threats, including connections with Denial of Service attacks:

- Some of the attacks reported - namely, Injection, XML External Entities (XXE) and Insecure Deserialization - may be vulnerable when Web Services are implemented without proper security features.

- Denial of Service Attacks can be performed, for example, after an Injection to compromise a server. Attackers may use this feature, for instance, to transform the server into a zombie machine.

- If a target application is vulnerable to XXE attacks, it likely means that this application is vulnerable to Denial of Service Attacks as well.

- A perpetrator may include a potential endless file inside the code generating a XML Denial of Service Attack (X-DDOS), which will be further explained in subsection 2.3.7.

Denial of Service attacks may also be seen as a constant threat to these technologies. A study conducted by the Cloud Security Alliance in 2017 [80] disclosed the 12 most dangerous threats to Cloud Computing according to surveys and questionnaires made with a series of working group members, in order to indicate the importance of each specific concern to their Organization. The working group also analyzed the security concerns using the STRIDE threat model, which was developed by Microsoft to evaluate information security threats. The study presented that, besides Denial of Service being one of these threats, it co-relates with six other threats from those twelve. This means that DoS could be connected to: Credential and Access Management, Insufficient Identity and insufficient Due Diligence threats; Specific system vulnerabilities, the Highjack of accounts, cause Abuse and nefarious use of Cloud Services and Data Loss.

The company Arbor Network, releases annually the Worldwide Infrastructure Security report. In their latest report [81] it revealed that DDoS attacks represent the dominant threat observed by the majority of service providers. DDoS and botnets are top security concerns for operators, even of IPv6-enabled networks. They also displayed that, the most popular targets of Application Layer Attacks were HTTP and DNS protocols. Both can be exploited using different variations of DDoS attacks. Application Layer DDoS attacks are seen at top three according to the report, which is an attack that targets specifically aspects of an application or service at the Application Layer. Although DDoS Volumetric attacks are still largely used and may also target the HTTP protocol, the use of DoS Low-Rate variant is increasing. They are most sophisticated and stealthy attacks compared to volumetric ones, because they can blend into legitimate traffic, be more effective using less computer power and generate traffic at a low rate.

Other particular concerns may be new forms of extortion. As Denial of Service Attacks may be used as smokescreen for other malicious activities. A study made by Yu [24] and also presented by Mather et. al. [48], disclosed that if the billing mechanism for cloud customers is pay-as-you-use, botnet owners can create a large number of fake users to intensively consume the service of the targeted cloud customer. DoS attacks on pay-as-you-go Cloud applications will result in a dramatic increase in the Cloud utility bill. In addition, a recent article by Microsoft about Cloud Threats [82] shows that Denial of Service for Ransom may become a common attack against Cloud Service Providers and Cloud-based Organizations. To stop a Cloud-dependent business, attackers may disrupt access to the Cloud instead of directly disrupt the Cloud Service Provider, and then hold the company for ransom. The article also revealed an increase in the number and variety of Internet of Things devices, meaning that this event may fracture some Cloud Security models, leading to successful attacks through these devices.

DDoS attacks are becoming larger and more frequent, and they are also becoming more sophisticated as they pinpoint specific applications or a weak point in the system design of the target.

Although sophisticated DDoS attacks require more understanding of the target system, some of them usually use less traffic and are harder to detect [83]. The Low-Rate Denial of Service attacks may be considered an example. Being the focus of this work, this threat will be explained in details in the section 2.4.

The access to Cloud Services are often delivered through the HTTP protocol. This means that HTTP protocol attacks, vulnerabilities, misconfiguration, and bugs have direct impact on the services deployed on the Cloud [84]. These services rely on the service providers for protection from threats, such as DoS Attacks, given their multi-tenant nature. Collateral damage, where attacks targeting one client impact another unintended client, represents a significant risk to all the customers of a Cloud Service Provider. An attack on one customer may potentially impact others [81]. The Low-Rate Denial of Service attack types aims mostly the application layer, therefore this threat may be considered critical to Cloud Services.

2.3 Distributed Denial of Service Attacks

2.3.1 DDoS Introduction and Concepts

A cyber attack by a malicious party aiming to disrupt a website on the Internet is called an availability-based attack. Using a wide spectrum of different attack vectors, availability-based attacks is one of the most serious security threats affecting websites. They are commonly referred to as Denial of Service (DoS) attacks. When the attack is carried out by more than one attacking machine, it is called a distributed denial-of-service (DDoS) attack [85].

DDoS is a coordinated attack, launched using a large number of compromised hosts. At an initial stage, the attacker identifies the vulnerabilities in one or more networks for installation of malware programs in multiple machines to control them from a remote location [86].

Every organization with a website – especially one that requires its users to have regular access to sensitive information – should take urgent and appropriate steps to protect against DoS and DDoS attacks. Failure to do so can result in huge financial losses as well as a damaged public reputation [85].

2.3.2 DDoS Characteristics and Motives

The first characteristic of this attack is to send illegitimate traffic to its victim, rather be an application, server, service or network, in order to overwhelm it. Another important one, is that Distributed Denial of Service creates its own network, using specific but known protocols to communicate with the *botnet* computers or devices. Other characteristic that may be observed, is the ability to attack a vast quantity of technologies, such as different protocols, Operating Systems, hardware equipment and devices the attack aims to. The amount of DDoS attacks variables is maybe the most adaptable feature this type of threat has. The next sections will show some examples of these types of attacks and briefly comment them.

Denial of Service attacks are normally executed with specific tools and scripts, in which numbers of packets are sent to a destination server - usually Web, FTP or e-mail servers - in order

to deny access to them. The Attack literally floods the servers resources making the system inoperative. This attack make computer systems inaccessible and do not only flood servers, but also networks and even specific user systems with useless traffic so that legitimate users can no longer gain access to those resources [87]. When this attempt derives from a single host of the network, it constitutes a Denial of Service attack. It is also possible that a several malicious hosts coordinate to flood the victim with an abundance of malicious packets, so that the attack takes place simultaneously from multiple points. This variation is called a Distributed Denial of Service attack [11].

Motives for DDoS attacks can have a variety of reasons. The 2017 Worldwide Infrastructure Security Report from Arbor Networks, showed in [81] that the top three reasons were (I) Criminals demonstrating web capabilities, where there is an indicative of the continuing weaponization of DDoS attacks via easy-to-procure services. (II) Online gaming-related and (III) gambling-related attacks, in which usually the cause is frustrated customers [88]. Criminal extortion attempts are also further on the list, as an example, in 2010 was revealed that one major Tokyo firm had to pay millions after the network was brought to a screeching halt by a botnet attack [35]. Other motives that are included in the reports are political and ideology disputes, usually when attackers shut down government web sites to stop their services. As examples, the stop of Rio 2016 Olympics web services in order to protest against the massive corruption in Brazil [89]; The Clinton and Trump Campaign web sites [89]; The 2009 Iran Elections protests, which were attacked by voluntary Low-rate DDoS [19], the Japanese Government Low-Rate DDoS Attack [90] and the Online Census of the Australian Bureau of Statistics [80].

2.3.3 DDoS Architecture and Attack Life-Cycle

A Distributed Denial of Service attack may be considered generally hierarchical. The DDoS network include one or more attacking hosts, numerous handlers and a vast number of agents. According to studies reported in works such as [88, 4] the architecture or strategy of a DDoS, as demonstrated in figure 2.5 can be divided in four parts:

- **Attacker:** Represents the perpetrator, the real attacker.
- **Handlers or Masters**: Attackers take control of unprotected devices in order to used them to scan other vulnerable devices to make them become *zombies* or, using a more technical name, *botnets*. These devices wait for the order from the attacker to do the scan.
- **Botnets or Zombies**: This *botnets* are packed with DDoS programs (usually *malware* and wait for the order from the handlers respecting the chain of command above to attack.
- **Target**: Any personal or network device, server or service/application being targeted by the DDoS attacker.

According with [91, 4, 86] The DDoS Life-Cycle is divided in four steps:

- **Selection of Agents**: The master attacker chooses the agents that will perform the attack. Based on the nature of vulnerabilities present, some machines are compromised to be used as agents [86]. The idea is to build an attack network which is distributed and consists of thousands of compromised computers [92].

- **Compromise/Exploitation and Expansion**: In order for attackers to create large botnets of computers under their control (referred to colloquially as zombies), they have two options: the more common option of using specialized *Malware* to infect the machines of users who are unaware that their machines are compromised, or the relatively newer option of amassing a large number of volunteers willing to use DoS programs in unison [85].
- **Communication/Command and Control**: The attacker communicates with any number of handlers to identify which agents are up and running, when to schedule attacks, or when to upgrade agents. Such communication among the attackers and handlers can be via various protocols such as ICMP, TCP, or UDP. Based on the configuration of the attack network, agents can communicate witha single handler or multiple handlers. [86].
- **Attack**: The master attacker initiates the attack. The victim, the duration of the attack, as well as special features of the attack such as the type, the length of TTL, and port numbers can be adjusted [86]. Although most DDoS attacks work via exhausting resources, the actual target which the service is going to be denied varies. The target could be a server application, where the attack disables legitimate clients to use a particular service. The target could also be the network access, where it disables the access to the victim computer or network by crashing it or overloading its communication mechanism or the network infrastructure [92].

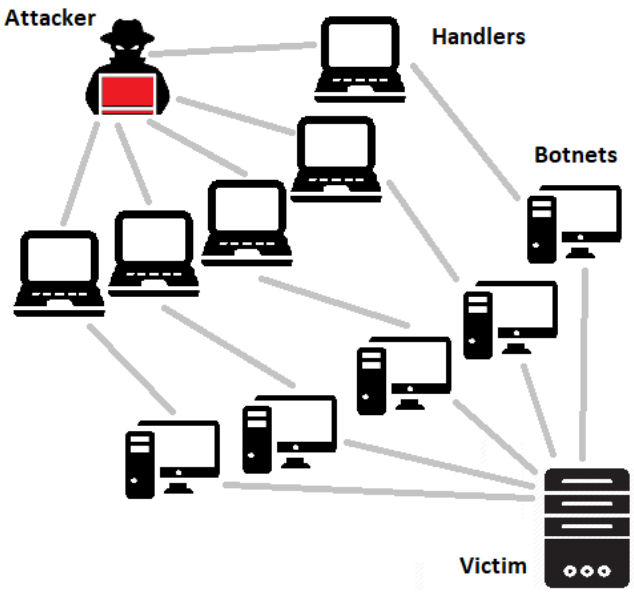

Figure 2.5: Distributed Denial of Service Attack Network

2.3.4 DDoS Categorization and Classification

In the literature, there are some works which try to categorize DDoS attacks [93, 92, 85, 94]. However, since there is no regulated standardization, in this work, the categorization of DDoS attacks is proposed based on three general categories according to Cisco and Arbor Networks [91, 95], of which some of them, also derive from the previous mentioned book and papers respectively [85, 94]. The taxonomy is based on these three categories and is distributed systematically:

- **Infrastructure Based**: It is the effort to absorb the bandwidth of the desired infrastructure target, such as network and servers. The attackers typically flood the victim with a high volume of packets or even connections. Volumetric attacks use an increased attack footprint that seeks to overwhelm the target. They generally use botnets to amplify this attack footprint. [91]

  An attacker may try to exhaust the hardware resources of network devices. For instance, if a router or firewall runs a certain set of packet filtering and logging rules, an attacker could bombard the device with packets that the router must filter out and log in an attempt to bog down the device [94].

- **Protocol/Operating System Based**: Attacks that target server resources attempt to exhaust a server processing capabilities or memory, potentially causing a denial-of-service condition. The idea is that an attacker can take advantage of an existing vulnerability on the target server (or a weakness in a communication protocol) in order to cause the target server to become busy handling illegitimate requests so that it no longer has the resources to handle legitimate ones. [85].

- **Application Based**: This category of the attack can target many different applications. However, the most common targets HTTP aiming to exhaust or disrupt Web servers and services. Some of these attacks are characteristically more effective than others because they require fewer network connections to achieve their goal [91].

  Instances of DDoS attacks that target application resources have grown recently and are widely used by attackers today. They target not only the well-known Hypertext Transfer Protocol (HTTP), but also HTTPS, DNS, SMTP, FTP, VOIP, and other application protocols that possess exploitable weaknesses allowing for DDoS attacks [85].

To classify the kinds of DDoS attacks may be difficult. Specially because of the amount of variations this threat has. In order to try a classification, DDoS should not be seen as a one dimensional attack. In the literature, There are some attempts to classify DDoS attacks, but just as in categorization, there are also no standardized approach to follow. Some classifications may be more used than others. For instance, for the purpose of comparison, the DDoS classification presented by Mirkovic et al. in [93] may have suggested a detailed classification of points. While the one proposed by Specht and Lee in [96], may have offered a simpler approach. For this work, the classification method used is an adaptation from the classification suggested by the two works aforementioned, in addition with works from Douligeris & Mitrokotsa [4] and Bhardwaj et al.[5]. The main purpose is to organize the general types of DDoS attacks which will be explained in the next subsection 2.3.5 and try to achieve a classification where may combine efficiency and a straightforward approach, as displayed in figure 2.6 and explained below:

- **by Degree of Automation**: The manual attacks involve the attacker scanning the network, IP Addresses, machines for vulnerabilities, break into the system and deploy a code and executes a malicious payload for remote control access of that user system which is kept ready to launch an attack on the attackers command [5].

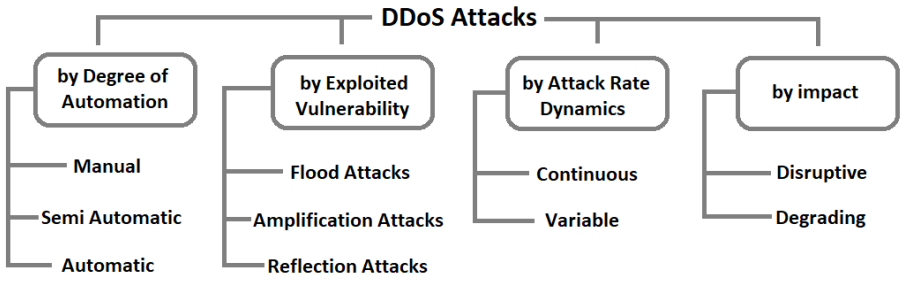

Figure 2.6: Distributed Denial of Service Classification (Adapted from [4, 5])

In the semi-automatic attacks, the attacker scans and compromises the handlers and agents by using automated scripts. The attack type, the victim address and the onset of the attack are specified by the handler machines [4].

The attack may also be automated, where the communication between attacker and agent machines is completely avoided. In most cases the attack phase is limited to a single command. All the features of the attack, for example the attack type, the duration and the victim address, are preprogrammed in the attack code. This way, the attacker has a minimal exposure and the possibility of revealing his identity is small [4].

- **by Exploited Vulnerability**: A flood attack involves zombies sending large volumes of traffic to a victim system, to congest the victim system's network bandwidth with IP traffic. The victim system slows down, crashes, or suffers from saturated network bandwidth, preventing access by legitimate users [96]. Flood attacks may be the predominant form of Denial of Service Attacks and some examples of flood-based attacks are the ICMP Flood and UDP flood.

  An amplification attack involves the attacker or the zombies sending messages to a broadcast IP address, to cause all systems in the subnet reached by the broadcast address to send a reply to the victim system. The broadcast IP address feature is found on most routers. When a sending system specifies a broadcast IP address as the destination address, the routers replicate the packet and send it to all the IP addresses within the broadcast address range. [96].

- **by Attack Rate Dynamics**: In the Continuous attack, after the onset is commanded, agent machines generate attack packets at a steady rate, usually as many as the agent machine resources permit. This approach provides the best cost-effectiveness to the attacker since it can be deployed with a minimal number of agents to inflict the damage. On the other hand, the large, continuous traffic stream can be detected as anomalous and arouse suspicion in the network hosting an agent machine, thus facilitating attack discovery. Variable rate attacks vary the attack rate of an agent machine to delay or avoid detection and response. [93].

- **by Impact**: The goal of disruptive attacks is to completely deny the victim's service to its clients [93]. Recovery from such disruptive attacks has the impact based on automated self-healing recovery, Human intervention and non-recoverable. Degrading attacks consume the victim resource gradually, in small portions. This makes the attack difficult to detect. [5]

2.3.5 DDoS General Types against Network Protocols

Many modern attacks typically use multiple vectors in a single attack campaign, targeting multiple components of a network of an organization infrastructure and its applications. Types of attacks include those that target network resources, those that target server resources, and those that target application resources [85]. In this subsection, some general and traditional types of Distributed Denial of Service attacks are explained. Since, they use known network standards and protocols to perform, these network knowledge are also briefly described according to each specific attack type.

- **TCP SYN Attack**: The Transmission Control Protocol specifies the format of the data and acknowledgements that two computers exchange to achieve a reliable transfer, as well as the procedures the computers use to ensure that the data arrives correctly [47].

  To establish a connection, TCP uses a three-way handshake. The first segment of a handshake can be identified because it has the SYN bit set in the code field. The second message has both the SYN bit and ACK bits set, indicating that it acknowledges the first SYN segment as well as continuing the handshake. The final handshake message is only an acknowledgement and is merely used to inform the destination that both sides agree that a connection has been established [47].

  Now that the functionality of TCP was briefly explained, the attacking system within the SYN attack, sends SYN request with spoofed source IP address to the victim host. These SYN requests appear to be legitimate. The spoofed address refers to a client system that does not exist. Hence, the final ACK message will never be sent to the victim server system [97]. This means that this type of attack takes advantage of how the implementation of TCP protocol works. They typically misuse the control bits (or flags) of the TCP protocol in order disrupt the normal mechanisms of TCP traffic [85]. When a connection is established, a port opens and it will only close until this connection finishes. The attacker tries to organize a large number of *botnets* to ensure that the port stays open and receive no more connections from legitimate users. As a result, with enough DDoS traffic, the server becomes unavailable.

- **UDP flood**: The User Datagram Protocol (UDP) provides an unreliable connectionless delivery service using IP to transport messages between machines. It uses IP to carry messages, but adds the ability to distinguish among multiple destinations within a given host computer. An application program that uses UDP accepts full responsibility for handling the problem of reliability, including message loss, duplication, delay, out-of-order delivery, and loss of connectivity [47].

  An UDP Flood attack consists of sending a large number of UDP datagrams from potentially spoofed IP addresses to random ports on a target server. The server receiving this traffic is unable to process every request, and consumes all of its bandwidth attempting to send ICMP `Destination Unreachable` packet replies to confirm that there was no application listening on the targeted ports [85]. This attack does not take advantage of a flaw in the protocol, instead it uses the usual behavior of UDP at such a point, that overburdens a specific network by creating large amount of requests.

- **ICMP Flood**: ICMP stands for Internet Control Message Protocol and is basically an error reporting protocol implemented within the Internet Protocol. The Internet Protocol is not designed to be absolutely reliable. The purpose of these control messages is to provide feedback about problems in the communication environment but There are still no guarantees that data will be delivered or a control message will be returned [98].

  An ICMP flood attack is a bandwidth attack that uses ICMP packets that can be directed to an individual machine or to an entire network. When a packet is sent from a machine to an IP broadcast address in the local network, all machines in the network receive the packet, which are going to reply back to the machine that sent. When a packet is sent from a machine to the IP broadcast address outside the local network, the packet is delivered to all machines in the target network [99].

- **HTTP Flood**: The Hypertext Transfer Protocol is a stateless to application-level protocol for distributed, collaborative, hypertext information systems. It was was created for the World Wide Web architecture and has evolved over time to support the scalability needs of a worldwide hypertext system. This protocol operates by exchanging messages across a reliable transport or session layer connection. An HTTP client is a program that establishes a connection to a server for the purpose of sending one or more HTTP requests [100]. The stateless characteristic mentioned, means that each HTTP request is self-contained; the server does not keep a history of previous requests or previous sessions [47].

  HTTP flood consists of what seem to be legitimate, session-based sets of HTTP GET or POST requests sent to a victim Web server, making it hard to detect. HTTP flood attacks are typically launched simultaneously from multiple computers [85]. While GET transfer a current representation of the target resource, POST perform resource-specific processing on the requested payload [101].

  Attackers generate a large number of valid HTTP requests (with GET/POST) to a victim web server. Despite not needed, attackers usually employ botnets to launch these attacks a successfully, since every single bot can generate a large number of valid requests. These are the most common HTTP Flood attacks, but other variations can be seen in [102].

- **Amplification and Reflection Techniques**: The attacker or the agents exploit the broadcast IP address feature found on most routers to amplify and reflect the attack and send messages to a broadcast IP address. This instructs the routers servicing the packets within the network to send them to all the IP addresses within the broadcast address range. This way the malicious traffic that is produced reduces the victim systems bandwidth. In this type of DDoS attack, the attacker can send the broadcast message directly, or by the use of agents to send the broadcast message in order to increase the volume of attacking traffic [4].

  According to the Worldwide Infrastructure Security Report [81], DNS is a popular target of application-layer attacks. Domain Name System is a directory `lookup` service that provides a mapping between the name of a host on the Internet and its numerical address

[87]. DNS has two, conceptually independent aspects. The first is abstract: it specifies the name syntax and rules for delegating authority over names. The second is concrete: it specifies the implementation of a distributed computing system that efficiently maps names to addresses [47].

While the most common form of this attack observed by US-CERT [103] involves DNS servers configured to allow unrestricted recursive resolution for any client on the Internet, these attacks can also involve authoritative name servers that do not provide recursive resolution. The primary technique consists of an attacker sending a DNS name lookup request to an open DNS server with the source address spoofed to be the target's address. When the DNS server sends the DNS record response, it is sent instead to the target. Attackers will typically submit a request for as much zone information as possible to maximize the amplification effect [104].

A reflector is any IP host that will return a packet if sent a packet. So, web servers, DNS servers, and routers are reflectors, since they return SYN, ACK or RST in response to SYN or other TCP packets. An attacker sends packets that require responses to the reflectors. The packets are ad dress-spoofed with source addresses set to a victim address. The reflectors return response packets to the victim according to the types of the attack packets. The attack architecture of reflector attacks consists in: A set of predetermined reflectors, these reflectors could also be dispersed on the Internet and The reflected packets are normal packets with legitimate source addresses and cannot be filtered based on route-based mechanisms [4].

2.3.6 DDoS Traditional Defense Techniques

There are approaches that are used especially by companies which develop infrastructure components. Usually, they may be used on one or more different techniques and equipment to try to respond against Distributed Denial of Service attacks. Although some of them can be effective against specific types of DDoS, others are not sufficient against more sophisticated attacks. Examples of these approaches can be found in literature such as [91, 85, 105, 106, 107].

In this subsection, traditional DDoS defense techniques are studied. Explanation about tools, tactics and implementation systems commonly used to mitigate Distributed Denial of Services attacks are explained:

- **Access Control List (ACL)**: A mechanism that implements access control for a system resource by enumerating the system entities that are permitted to access the resource and stating, either implicitly or explicitly, the access modes granted to each entity [108].

  As mentioned throughout the literature [93, 95, 85], due to the amount of botnets and large traffic generation related to Distributed Denial of Service attacks, once an attack is identified, the immediate response is to identify the attack source and block its traffic accordingly. The blocking part is usually performed under manual control since an automated response system might cause further service degradation in response to a false

alarm [4]. Therefore, Successful access control requires a careful combination of restrictions on network topology, intermediate information staging, and packet filters [47].

- **Conventional Firewalls**: A Firewall act as the first line of defense against unwanted and malicious traffic and also represents critical point of failure during DDoS attack [109]. A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks [87].

  Firewalls keep track of the status of all network connections that they inspect. All such connections are stored in a connection table, and every packet is matched against that connection table to verify that it is being transmitted over an established legitimate connection [85]. Its functions can be diversified and as Stallings disclosure in [87], Types of Firewall can be: Packet Filtering Firewall, Stateful Inspection Firewalls, Application-Level Gateway and Circuit-Level Gateway.

  In the absence of a dedicated anti-DoS device to shield the firewall from such a high volume of traffic, the firewall itself is usually the first device in an organization network to handle the force of the DDoS attack. Because of the way a firewall is designed, it will open a new connection in its connection table for each malicious packet, resulting in the exhaustion of the connection table in a very short period of time [85].

- **Manual Reactions and Blackholing**: Manual responses to DDoS attacks focus on measures and solutions that are based on details administrators discover about the attack. For example, when an attack such as an HTTP GET/POST flood occurs, given the information known, an organization can create an Access Control List to filter known *bad actors* or bad IPs and domains [91]. Another option may be considered a more drastic approach, is Blackholing. This method consists in detours all traffic coming from previous identified attack sources to a *black hole*. Usually, this is used by changing the packets that are arriving from these sources to a `Null` path. The use of null routing, or blackholing, also increased to counterattack such issues, but it is not a viable or acceptable longterm strategy [110]. Similar to other solutions previously mentioned, this also may cause the victim deprived from legitimate traffic.

- **Intrusion Detection Systems (IDS)**: Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified. Intrusion detection systems have been developed to provide early warning of an intrusion so that defensive action can be taken to prevent or minimize damage [87].

  In classical physical enterprise settings, an IDS is normally deployed on dedicated hardware at the edge of the defended networking infrastructure, in order to protect it from external attacks. In a cloud computing environment, where computing and communication resources are shared among several users on an on-demand, pay-per-use basis, such strategy is not effective: attacks may be originated within the infrastructure itself [41].

  As Liao stated in [111], Intrusion Detection methodologies are classified as three major categories: Signature-based Detection, Anomaly-based Detection and Stateful Protocol Analysis (SPA):

- **Signature-based Detection**: A signature is a pattern or string that corresponds to a known attack or threat. SD is the process to compare patterns against captured events for recognizing possible intrusions [111]. It relies on the pre-installed intrusion patterns or strings on the database. This operating manner limits its performance because the malware sig- nature will not be detected when the Signature Detection system is not updated [112]. Signature Detection is also known as *Knowledge-based Detection* or *Misuse Detection* [111].

- **Anomaly-based Detection**: An anomaly is a deviation to a known behavior, and profiles represent the normal or expected behaviors derived from monitoring regular activities, network connections, hosts or users over a period of time. Then, AD compares normal patterns of behaviors with observed events to recognize significant attacks [111]. The comparison can determine whether there is a partition between normal and unusual behaviors, and the unusual behavior is considered as an active or potential attack, which depends on the level of differences [112]. Anomaly Detection is also called *Behavior-based Detection* [111].

- **Stateful Protocol Analysis**: The *stateful* indicates that IDS could know and trace the protocol states (e.g. pairing requests with replies). It depends on vendor-developed generic profiles to specific protocols. Stateful Protocol Aanalysis is also known as *Specification-based Detection* [111]. The main positive feature of this technique is providing protocol analysis with stateful characteristics but this technique does not completely solve the problem of detecting attacks based on a single request or response. Addressing this problem, more stateful characteristics need to be added to the protocol analysis profile, which synchronously causes heavy workloads and oversized packets [112].

- **Intrusion Prevention Systems (IPS)**: Intrusion prevention is performed by a software or hardware device that can intercept detected threats in real time and prevent them from moving closer toward victims. It monitors network traffic and system activities for malicious activity; however, unlike an IDS, an Intrusion Prevention System is able to actively block intrusions that are detected. Typically, an IPS does so by generating alarms, dropping malicious packets, resetting the connection, and/or blocking traffic from the offending IP addresses [86].

  Intrusion Detection enables the collection of information about intrusion techniques that can be used to strengthen the Intrusion Prevention facility [87].

Traditional security systems were designed to prevent intrusions that would lead to a data loss or breaches. By having devices perform full session inspections and network firewalls to enforce policy on user traffic, it helped determine if the traffic needs to be allowed into the data center as per predefined rules [40]. Also, most DDoS defense mechanisms are deployed at the victim end for effective detection and defense of a system. Victim-end detection systems detect attacks either in a reactive or proactive manner. A victim-end defense approach cannot provide complete protection from DDoS attacks when the rate of attack traffic is very high. The DDoS attack may have already made damage on the target [86]. Therefore, one may conclude that the traditional defense techniques studied in this subsection may be outdated against advanced DDoS variants.

2.3.7 DDoS Advanced Variants and Countermeasures

The availability of the sophisticated tools and other resources to perform a DDoS attack is becoming more plentiful. As a result, the frequency and the power in terms of complexity and volume of DDoS attacks are increasing [86]. In this subsection, more refined Distributed Denial of Service attacks are disclosed. As well as, advanced techniques which try to counter DDoS threats.

Different from the traditional types, these advanced DDoS variants do not use volumetric floods to deny access to legitimate users. Instead, they use different approaches, attacking other protocols and technologies. These advanced attacks are explained below:

- **Low-Rate Attacks**: DDoS attacks can also be divided into High-Rate and Low-Rate attacks. In the high rate attack scenario the victim is flooded by a large number of HTTP requests. While, in the Low-Rate attack scenario slow and compromised requests are used to engage the victim in high complexity computations which consume a significant amount of its resources. As there is no illegitimate TCP or UPD packets in these attacks they easily avoid the Network layer detection techniques [84] discussed in 2.3.6.

  Since Low-Rate Distributed Denial of Service attacks is the focus of this work, in the next section 2.4 this threat will be fully analyzed in order to demonstrate its potential and its operation.

- **Distributed XML Denial of Service**: Distributed XML-based DoS (DX-DoS) occurs when an XML message is sent to a Web Server or Web Service with malicious content to use up all its resources. One example of an X-DoS attack is called *Coercive Parsing attack*, which manipulates the Web Service request when a Simple Object Access Protocol (SOAP) is parsed to it so that it can transform the content to make it accessible to applications. The Coercive Parsing attack uses a continuous sequence of open tags so that the CPU usage on a web server becomes exhausted [113].

  The most recent OWASP top 10 threats [79] presents that attackers can exploit vulnerable XML processors if they are able to upload XML or include hostile content in an XML document, exploiting vulnerable code, dependencies or integration. This happens because several bad configured XML allows external entities to interact within XML documents. This way, those entities can manipulate code inside XML messages to contain any information and benefit from this to create Denial of Service attacks using infinite looping codes.

- **SSL Denial of Service**: SSL-based DoS attacks take many forms: targeting the SSL handshake mechanism, sending garbage data to the SSL server, or abusing certain functions related to the SSL encryption key negotiation process. SSL-based attacks could also simply mean that the DoS attack is launched over SSL-encrypted traffic, which makes it extremely difficult to identify. Such attacks are often considered asymmetric, as it takes significantly more server resources to deal with an SSL-based attack than it does to launch one [85].

With new vector attacks and threats on the rise, corporations and enterprise are required to protect their IT infrastructure from the advanced attack methods being employed. In addition to an increase in frequency, attacks have also become more sophisticated and stealthy [40].

Since these attacks targets different protocols and even distinct layers of network model, when traditional defense techniques are paired with these advanced DDoS attacks, one can analyze that they may not be fully suitable to fight against this kinds of threats. For example, Application Layer attacks often consist of what appears to be legitimate traffic, making them more difficult to detect [40]. Considering these facts, some other techniques are being used to try to fight these advanced DDoS techniques, such as:

- **Web Application Firewall**: With the advance of Cloud Computing and its enable technologies such as Service-Oriented Architecture and Web Services, new technologies can be used in other manners such as in securing a system. Web Application Firewalls (WAF) may be an example of this statement. A web application firewall (WAF) is generally hardware and/or software that resides between a web client and a web server. WAF works applying a set of rules to Hypertext Transfer Protocol (HTTP) conversations. Generally, the rules applied by the WAF aim to prevent common attacks such as Cross Site Scripting (XSS) and Structured Query Language (SQL) injection [114]. Some studies such as [40, 115], revealed to use WAF combined with Content Delivery Networks (CDN) as DDoS defense for a Cloud Environment. WAF can also be applied to parts of the infrastructure related to data traffic (e.g. between web server - physical firewall - network equipment). Unlike traditional firewalls, Web Application Firewalls can terminate SSL traffic and work after the data is decrypted.

  ModSecurity may be cited as an example of WAF. It allows users to enhance the security of servers by detecting and preventing attacks before they reach web applications [116]. ModSecurity started as a module embedded into web servers but nowadays evolved to become a standalone system and one of the primary choices as a Web Application Firewall. ModSecurity grant users to prevail against application DDoS related threats without changes in the current infrastructure. It is Flexible, uses regular expressions based rule engine and allows rules to be combined. Support unlimited number of different policies (e.g. per file, per folder, per virtual host, etc.) and includes audit logging.

- **Web Server Modules**: Modules are extensions that enhance the basic functionality of the Web server. The modules reflect the growth of the Web and the inclusion of dynamic content into the web pages [116].These modules may help to mitigate some advanced DDoS techniques:

  - **Apache mod_evasive**: The mod_evasive Apache module is a popular security solution that provides a measure of protection against application layer Denial of Service attacks. It works by inspecting and verifying incoming traffic to an application server using a dynamic hash table of IP addresses and URLs. Essentially, mod_evasive is a rate limiting solution that blocks traffic from IPs that exceed a predetermined threshold for the number of requests to a specific URI or domain [117]. Its disadvantage, is that despite mod_evasive ability to detect some application-based attacks, it will not warn when a low-rate type attack occur, because will not be able to differentiate

legitimate from malicious traffic.

- **Apache mod_qos**: It is a quality of service module for the Apache web server which implements control mechanisms that can provide different levels of priority to different HTTP requests. Quality of service implements control mechanisms to provide different priority to different users, applications, and data connections. It is used to guarantee a certain level of performance to data resources [118].

- **Apache mod_reqtimeout**: This module provides a convenient way to set timeouts and minimum data rates for receiving requests. Should a timeout occur or a data rate be too slow, the corresponding connection will be closed by the server. Can use the directive `RequestReadTimeout` to set various timeouts for receiving the request headers and the request body from the client. For example, if the client fails to send headers or body within the configured time, a 408 REQUEST TIME OUT error is sent. In order to do that, the `header`, body and minrate can be configured to a chosen value and the directive can check all of them simultaneously [119]. Its negative aspect is that a simple modification in the strategy of the attack can bypass its preprogrammed values. For example, an attacker can send single requests pre-attack with different time intervals using different `header`, body and minrate values. This way, everytime the perpetrator chooses a value and analyze the response of the server - if the server drops a packet or if it received normally, the perpetrator will be able to deduce an approximate value configured in the directive. Therefore, the attacker can configure its script or tool to send requests with a value smaller but closer to the `header`, `body` and `minrate`, avoiding the module defense [20].

- **IIS/Azure Defense**: Microsoft has a Security Guideline, which can be seen in [107], to detect and prevent DOS attacks targeting IIS/Azure Web Role (PAAS). This guideline consists in best practices and hardening configurations that servers with their products must conduct to better the defense against Distributed Denial of Service Attacks. Basically they use several combined techniques such as log check tools, dynamic IP restrictions, IP reputation check, captcha and Sinckhole Mechanisms to achieve better results. Windows Azure has a specific defense system designed not only to withstand attacks from the outside, but also from inside. Azure DDoS protection also said to provide defense for application layer attacks.

- **Content Delivery Network (CDN)**: CDN deploy servers in various locations where they host proxies of websites, distributing computational and storage resources and delivering content to clients from a nearby server. CDNs use various mechanisms to provide scalable and robust services, including web-caches that reduce latency and communication volumes, and filters against clogging traffic [39].

  They were originally used to reduce the latency of web access by redirecting the user to a surrogate server (or cache server) close to the user, as well as to lighten the load of original web servers. In recent years, CDN providers also start to offer DDoS mitigation services by hiding the original web site and distributing the load of attack traffic to multiple surrogate servers [115].

One problem with CDNs is the high-price for using this technology since it uses several resources from a large infrastructure network. One reason for the limited use of this technology is that smaller Organizations may not be able to afford use CDNs on a regular basis, i.e., when not under attack, while temporal migration to a CDN introduces substantial administrative effort and financial costs [39].

- **Hardware Options**: Several IT infrastructure companies provide equipment to protect against threats. Traditional perimeter security is composed of static security controls. Network security devices are placed in network traffic aggregation points and on gateways. They are also put in the frontier of the inner perimeter and the outside environment [110].

  This hardware devices have specific DDoS protection systems embedded. For example, CISCO uses NetFlow [91] in several of its products to monitor DDoS traffic and partners with other network companies in order to develop solutions to defend all types of DDoS as a whole. This is done by integrating CISCO ACI with other companies defense systems such as, Radware Attack Mitigation System (AMS) and F5 Synthesis [85, 105]. This approach assumes a fixed network infrastructure, but that is not what is happening with nowadays infrastructure [110].

Enterprises that aspire to guarantee availability of online services are urged to prioritize specially designed DDoS attack mitigation solutions [40]. The proliferation of smartphones and portable computing devices allowed the emergence of the newly BYOD paradigm. This brings new classes of security issues that have an intrinsic relationship with mobile-enabled malware spread and an impact in cloud applications accessible though mobile platforms [110].

2.4 Low-Rate Distributed Denial of Service Attacks in the Application Layer

2.4.1 Low-Rate DDoS Introduction

As aforementioned in the previous section, new defense techniques are being used to counter Advanced DDoS attacks. DDoS attacks may have in common the fact that it uses high volume bandwidth to denial a specific server or service. As presented throughout the section 2.3.5, in order to achieve that, DDoS may use different variations which harm different network layers. Unlike flood attacks specified in the section above 2.3.7, Low-Rate attacks do not require a large amount of traffic. They target specific design flaws or vulnerabilities on a target server with a relatively small amount of malicious traffic, eventually causing it to crash [85].

2.4.2 Low-Rate DDoS Fundamental Concepts

A low-rate DDoS attack is an intelligent attack as the attacker can send attack packets to the victim at a sufficiently low rate to elude detection. Today, a large-scale DDoS attack is usually combined with multiple low-rate attacks, which are distributed on the Internet to avoid being detected by current detection schemes [22].

Low-Rate attacks can target the TCP Congestion Control Mechanism and TCP Retransmission Timeout by sending periodically pulsing data flows instead of sending continuous network traffic. This may dramatically reduce the average rate of attack flows [120]. LDDoS can also aim specifically the Application-Layer, as some literature presents [90, 121, 122, 123].

A study produced by Arbor Networks (fig W2) in 2016, displayed that DDoS attacks against the Application Layer are on the rise.
As presented in the literature [124, 90, 22, 125, 121, 126], Low-Rate Denial of Service attacks in the Application-Layer explores the communication between client and server using HTTP messages. The Characteristics of this threat may be cited as: **I)** It does not need computer power to be performed since it send small pieces of data through the network in period of times and **II)** considering its first characteristic, it can be mistaken with normal, legitimate traffic.

Since these variants of the LDDoS may abuse the HTTP protocol, it is important to briefly describe how the packet containing HTTP messages works. This way, the differences between a malicious Low-Rate attack packet and legitimate traffic packet can be noted.

2.4.3 The HTTP message Format

Accordingly with the RFC 7230, the message format of HTTP protocol [100] consist of a start-line, followed by a sequence of octets in a specific format, referred as `Header-Field`. After, comes an empty line indicating the end of the header section. Another empty line separating the Header with the message body, which can be optional. These empty lines are called Carriage Return Line Feed (CRLF):

```
HTTP-message = start-line
               *(header-field CRLF)
               CRLF
               [message-body]
```

An HTTP message can be either a request from client to server or a response from server to client. Syntactically, the two types of message differ only in the start-line, which is either a request-line, for requests or a status-line, for responses. The request-line begins with a method token, followed by a single space (SP), the request-target, another single space (SP), the protocol version, and ends with CRLF. The status-line of the response message, consists of the protocol version, a space (SP), the status code, another space, a possibly empty textual phrase describing the status code, and ending with another CRLF [100].

```
request-line = method SP request-target SP HTTP-version CRLF

status-line = HTTP-version SP status-code SP reason-phrase CRLF
```

In particular, focusing on the method token in the request line and the status-code in the status-line. The method token indicates the request method to be performed on the target resource. The request method is case-sensitive. e.g. GET, POST. The status-code is an element is a 3-digit integer code describing the result of the server's attempt to understand and satisfy the client's corresponding request. e.g. 400, 404, 500 [101].

The message body (if any) of an HTTP message is used to carry the payload body of that request or response. The presence of a message body in a request is signaled by a `Content-Length` or `Transfer-Encoding` header field. The presence of a message body is a response that depends on both the request method to which it is responding and the response status code [101].

2.4.4 Low-Rate DDoS Types and Operation Methods

Technically, basing on its characteristics, one may state that any type of attack can be done in a low-rate pace. But usually, Low-Rate attacks exploit the manner in which a protocol or application works. Below, are known and used variants of this type of attack. Each one has their own specific exploitation method:

- **Slow GET HTTP Attack (Slowloris)**: This attacks makes use of HTTP GET requests to occupy all available HTTP connections permitted on a web server and takes advantage of a vulnerability in thread-based web servers which wait for entire HTTP headers to be received before releasing the connection. While some thread-based web servers such as Apache make use of a timeout to wait for an HTTP request to complete, this timeout - which is set to 300 seconds by default - is re-set as soon as the client sends additional data [121, 122]. The picture 2.7, displays a depiction of this type of attack.

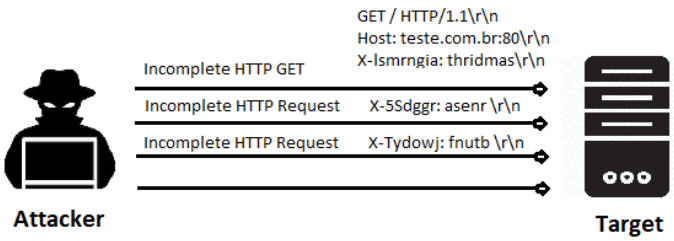

Figure 2.7: Slow HTTP GET (Slowloris)

This creates a situation where a malicious user (or various) could open several connections on a server by initiating an HTTP request but does not close them. By keeping the HTTP request open and feeding the server fake data before the timeout is reached, the HTTP connection will remain open until the attacker closes it. Naturally, if an attacker had to occupy all available HTTP connections on a web server, legitimate users would not be able to have their HTTP requests processed by the server, thus experiencing a Denial of Service [121].

Slowloris is a known tool to perform this attack - such as, this type of attack is also referred to as Slowloris Attack - It was first mentioned by Adrian Ilarion Ciobanu in 2007 and created

by Robert *RSnake* Hansen in 2009 and its main characteristics and weaknesses are [125, 122, 126]:

- Low Bandwidth
- Keep Sockets Alive
- Only Affects Certain Web Servers
- Does not work through Load Balancers
- Reverse Proxies can help mitigate the attack
- Is not anonymous by itself

When a client-server communication is made, the server opens a socket to receive the information from the client. When it is a HTTP GET request, the double CRLF field indicates the end of the message as explained above in 2.4.3. If a Slowloris attack in being executed, it sends the data and leaves out the last CRLF. This way, the Web server interprets that the message did not end. Leaving the socket open. This means that in a distributed attack, where several bots are used, the Web server can have its sockets filled with unfinished requests and as a result, it leaves no space for authentic users to access the Web server.

Hence, the Server returns several `404: Bad request` errors. This status code indicates that the server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing) [101].

Slowloris lets the web server return to normal almost instantly (usually within 5 seconds or so depending on the size of the infrastructure). That makes it is ideal for using it as a distraction and use other attacks that may just require a brief down-time [122]. Also, it works in pulses, leaving the services unavailable from time to time.

According to a survey made by Netcraft in July of 2017 [126] - a company that makes surveys of websites and hostnames used in the whole world since 1995, including web servers and computers use - Even though Microsoft now serves more than half of the hostname sites in the world, Apache is still the leading web server of active web sites. Having almost half of the world websites. As aforementioned, The Slow GET variant is efficient against thread-based servers like Apache. This makes countless servers from the world vulnerable to this attack.

- **Slow HTTP POST Attack**: The POST method requests that the target resource process the representation enclosed in the request according to the resource's own specific semantics. As an example, POST may be used for: Providing a block of data, Posting messages to bulletin boards, newsgroup, mailing list, blog, or similar group of articles and Appending data to a resource [101]. The important distinction between a GET request and a POST request is that a GET only has a header, while a POST has both a header and a body. Both header and body are terminated by a blank line, as aforementioned in the HTTP message forma 2.4.3.

In order to carry out a slow HTTP POST request attack, the attacker detects forms on the target Website and sends HTTP POST requests to the Web server through these forms. The POST requests, rather than being sent normally, are sent byte-by-byte. As with a slow HTTP GET request, the attacker ensures that his or her malicious connection remains open by regularly sending each new byte of POST information slowly at regular intervals. The server, aware of the content-length of the HTTP POST request, has no choice but to wait for the full POST request to be received (this behavior mimics legitimate users with slow Internet connection) [85]. A representation of Slow HTTP POST attack in described in the picture 2.8. This attack, similar to Slow HTTP GET attack, also works in pulses.

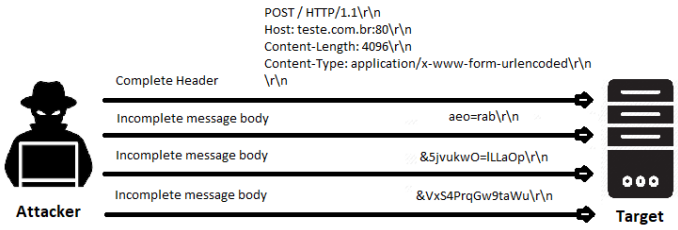

Figure 2.8: Slow HTTP POST (R.U.D.Y.)

A very traditional attack tool is known as *R-U-Dead-Yet?* or its shortened version, R.U.D.Y. Just as *Slowloris*, at the time of its creation, R.U.D.Y. delivered a very peculiar type of attack which had to fill in a classification. Therefore, it may not be unusual to see the Slow HTTP POST attack be called by the same name as this tool. R.U.D.Y. attacks take advantages of the fact that an HTTP POST operation allows for the connection to remain open indefinitely in cases where the POST data arrives very slowly. The low and slow traffic associated with this attack makes it hard for the traditional mitigation tools to detect. [127].

- **Slow HTTP Read Attack**: An attacker basically sends a legitimate HTTP request to target Web server and then very slowly reads the response. If a HTTP request is not complete, or if the transfer rate is very low, the Web server keeps its resources busy waiting for the rest of data [90]. An attacker may send legitimate requests to the server - simple communication accordingly to the three-way-handshake TCP procedure - and will disclosure to the server a `window size` field shorter than usual. This way, the HTTP server response will be slower since the size of the response packets will be reduced. This attack is illustrated in figure 2.9.

  There is a tool called SlowHTTPTest which is a configurable tool that simulates some Application-Layer Denial of Service attacks by prolonging HTTP connections in different ways. It implements most common low-bandwidth Application Layer DoS attacks, such as Slow HTTP GET, Slow HTTP POST and Slow HTTP Read attack by draining concurrent connections pool, causing very significant memory and CPU usage on the server [128].

Accordingly with Damon et. al. [129], HTTP is vulnerable for many of the same reasons that TCP was. Resources are allocated by the server when a request is initiated, and they remain allocated while the connection is open, regardless of whether the client is utilizing them or not.

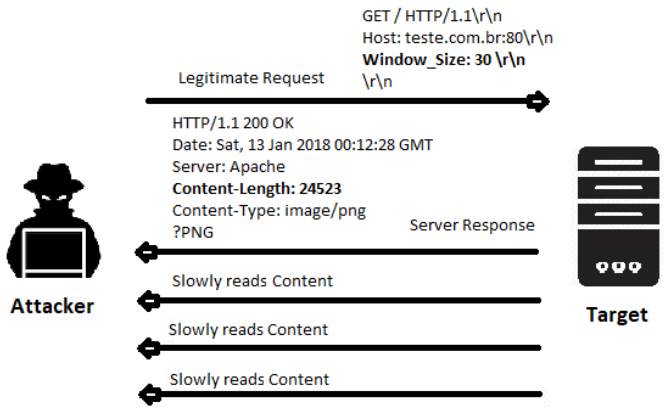

Figure 2.9: Slow HTTP READ

2.4.5 Existing Protections Against Low-Rate DDoS

As studied throughout this section 2.4, Low-Rate Distributed Denial of Service attack may be hard to detect compared to classic DDoS attacks. There are some detection method being used to prevent this attack to explore the HTTP protocol weaknesses. The Open Web Application Security Project [123], even suggests some mitigation methods which can be used against Low-Rate DDoS, using traditional Hardening Techniques, Protocol Configuration Controls and a traditional IDS tools which may be evolving to repent this threat as well. These mitigation methods are:

- Drop connections with abnormally small TCP advertised windows;
- Set an absolute connection timeout, if possible Limit length, number of headers to accept;
- Limit max size of message body to accept;
- Drop connections with HTTP methods (verbs) not supported by the URL;
- Limit accepted header and message body to a minimal reasonable length;
- Define the minimum data rate, and drop connections that are slower than that rate;
- Qualys Web Application Scanner passively detects the slow attack vulnerabilities;
- ModSecurity v2.6 introduced a directive called SecWriteStateLimit that places a time limit on the concurrent number of threads (per IP address);
- Snort is working on detecting connections with small TCP advertised windows;

With the rise of Cloud Computing, some companies such as Akamai [106, 13] and F5 Networks [105] gained more visibility with different techniques approaching DDoS vulnerabilities, especially because they try to mitigate advanced threats like Low-Rate DDoS. They said to use different approaches to secure Cloud Computing, exploring technologies like Web Application Firewall technology, Web Application Scanning and Application Delivery Network, which are explained below:

- **Web Application Firewall**: As mentioned in the section above 2.3.7, Web Application Firewalls (WAF), are also being used to work against advanced DDoS threats. Companies such as Akamai have a platform which uses WAF to help protecting its clients [106]. They claim to have protections against Low-Rate DDoS attacks specifically to each type, for Slow

Headers attacks their Edge servers always wait to receive the full headers from a client before it starts streaming the request forward to the origin Web server. For Slow POST, they keep track of the rate at which it receives the data from the client. It is possible to define the minimum bit rate and the number of intervals. The Edge server will wait before deciding that a client is too slow and is likely attempting an attack.

- **Web Application Scanning**: Web Application Scanning (WAS) technology can find and catalog all web applications inside a specific network. This way, it can create reports of vulnerabilities based on already knows threats. Regarding Low-Rate DDoS attacks, Qualys [130] use its WAS to opens two connections to the server and requests the base URL provided in the scan configuration in - in case of a Slow HTTP header attack - and opens two other connections, and uses an action URL of a form it discovered during the crawl phase that does not require authentication - in case o Slow HTTP POST attack. With those information, WAS considers the vulnerability based on some conditions as shown in.

- **Application Delivery Networks**: A proposed architecture by F5 Networks [105] using its already build IT infrastructure, make it possible to build an Application Delivery Network (ADN) that is DDoS-resistant based on a two-tier solution, where the first tier is a layer 3 and 4 network firewall type architecture and the second tier focus on application defense specifically against SSL attacks and HTTP related attacks.

- **Apache Specific Modules**: Apache has other module besides the ones cited in 2.3.7 which it is said to helps specifically against Low-Rate Distributed Denial of Service attacks, particularly against Slow GET HTTP type. It is called `mod_antiloris` and the recommended strategy is to combine it with the other modules in order to obtain better results.

  This module has a strategy in which it uses the counting of open simultaneous connections from the same IP address. When this counting is superior to the configured limit on the server (default value = 10), the module rejects all the requisitions made by that specific IP address [20]. This strategy may be considered rather aggressive since it can generate several false positives blocking real users from access the server. For example, if a company uses intranet and uses one or few public IP address, concurrent requests from inside that company will appear in the server as only one IP address. Also, it only try to mitigate one type of Low-Rate attack, being of no use against the other types.

2.4.6 Challenges in Detection and Mitigation of Low-Rate DDoS

The challenge in preventing Low-Rate DDoS attacks lies in the nature of the traffic and the nature of the attack itself. Therefore, there is not a straightforward approach or method to filter or block the offending traffic [91]. The technologies studied throughout this section may be static such as the Apache modules, therefore, not that effective. Other technologies are proprietary, hence they can be expensive or even change the architecture in which the company works on. Summarizing, they may not be a guaranteed solution do stop or mitigate Low-Rate DDoS attacks.

Tools such as Slowloris and R.U.D.Y. produce legitimate packets at a slow rate, allowing attacks carried out using them to pass traditional mitigation strategies undetected [92] and as presented in the literature [125, 121, 127, 129], to perform this attacks one does not require large resources from the the attacker computer. These tools are also available online for free, so anyone can download, use or even modify the scripts to suit a precise necessity. Even this type of attack being a difficult one to detect, some methods that researchers found to help identify and mitigate LDDoS, actually detect the presence of LDoS-like attacks but do not necessary distinguish the specific flows associated with it. More detailed information about it in section 2.5.

Today, a large-scale DDoS attack is usually combined with multiple low-rate attacks, which are distributed on the Internet to avoid being detected by current detection schemes [22]. As detection methods provided in the researches are concentrated on packet-level discovery [131, 120], there are fewer works which focus on flow-level detection and mitigation of Low-Rate DDoS itself such as [84]. Packet-Level inspection methods may be more accurate but are also proved to be more resource consuming then Flow-level systems, since they check for every single packet that passes through a network card.

Tests of Low-Rate DDoS attack may be difficult due to the necessary infrastructure to do so. Most of the tests at application levels are done using datasets, which are a basically a collection of traffic information gathered during a period of time using specific computer programs and scripts to listen network cards, capture those packets and saving it as an archive. To create a IT infrastructure to actually generate legitimate traffic as well as Distributed Denial of Service malicious traffic in real time and in large scale is uncommon among the literature and uneasy to replicate.

With Cloud Computing infrastructure size, it also became simpler to amplify a DDoS attack. Using VMs, devices and equipment as reflectors for a larger attack. Despite these Data centers have great security, they also receive so much more requests than traditional in-door data centers. Due to the large amount of information circulating inside Cloud structures, dealing with Low-Rate Distributed Denial of Service attack to distinguish real user traffic from malicious traffic may become even harder. These attacks should be tempered and have almost or no effect to the accesses of clients to the services providers.

2.5 Related Research on Low-Rate DDoS Detection and Mitigation Methods

Despite some new defense structures presented some results in mitigating Low-Rate Distributed Denial of Service attacks, these systems may be complex, usually involve the creation or use of other infrastructure approaches and the ones dealing directly in the application layer, may lack early detection. Also there are not enough defense mechanisms specifically for this attack and traditional DDoS defense techniques, as studied throughout this work, to help. Based on the severity of this attack, some efforts are being made by researchers to develop more efficient techniques to detect and mitigate Low-Rate Distributed Denial of Service attacks in general. In this section, researches made on the subject are going to be explored and analyzed with the purpose of extract more detailed information regarding the amount of detection methods for

Low-Rate DDoS attacks and briefly present their effectiveness according to the authors.

Z. Liu and L. Guan [124] proposed a scheme were LDoS attack traffic was simulated on OPNET platform and generated by estimating the re-transmission time out (RTO) of normal TCP flow based on the analysis of LDoS attack features, and the performance of targets under the attacking of LDoS was tested. Their method consisting investigate the cache queue of target router for the purpose of detection and defense of LDoS attack. Test result show that two criteria – packet percentage and threshold – can detect LDoS attack flows: Attack simulation and signature extraction of low-rate DoS. They consider peak moments of a Low-Rate DDoS attack compared to legitimate traffic. The length of these peaks is considerably small compared to legitimate traffic, inside a specific time window. To study the effectiveness of their method, they performed simulation experiments to verify the attacked server's responses comparing the peak of legitimate traffic with Low-Rate DDoS traffic.

Xiang, Li and Zhou [22] suggests a different method using Generalized Entropy and Information Distance. The approach consists in detect Low-Rate attacks earlier and more accurately which are two important criteria for the success of the defense system. The proposed detection systems can use either the source IP address-based method or the IP packet size-based method to calculate the probability distribution of the traffic in the given time interval. The IP packet size-based method is to utilize the feature that attacks usually produce packets in defiance of a victim's response and when a flooding-based attack occurs, the same sized packets are generally used. On the other hand, the legitimate network traffics have typical packet sizes with respect to requests and responses or data and acknowledgments. The tests were performed using differences between legitimate and malicious traffic datasets and the results were that the method works effectively and stably outperforming traditional Shannon Entropy and kullback-leibler. They also created a scheme to traceback the attackers and identify zombie machines.

In a article by C. Zhang et al. [120] propose a novel metric – Congestion Participation Rate (CPR) – and a CPR-based approach to detect and filter Low-Rate DDoS attacks by their intention to congest the network. The considered innovation of the CPR-base approach at the time was its ability to identify Low-Rate DDoS flows. The authors explain that a flow with a CPR higher than a predefined threshold is classified as an Low-Rate DDoS flow, and consequently all of its packets are be dropped. To validate their results and evaluate the performance of the pro posed approach, they conduct ns-2 simulations, test-bed experiments, and Internet traffic trace analysis. They compare their work with Discrete Fourier Transform based approach which is considered one of the most efficient approaches in detecting Low-Rate DDoS attacks. The experiment results have demonstrated that, compared to the existing DFT-based approach, the CPR-based approach is effective for all Low-Rate DDoS attacks considered, while the DFT-based approach is effective for a limited set of Low-Rate DDoS attack types. Despite their project intended to mitigate TCP level Low-Rate Distributed Denial of Service at the time, it is tested alongside with HTTP flows and packets generated by a tool that uses real Internet traces.

In [132] Bhuyan et al. propose an Empirical Evaluation where they analyze Hartley entropy, Shannon entropy, Renyi's entropy, Generalized entropy, Kullback-Leibler divergence and Generalized information distance measure in their ability to detect both low-rate and high-rate DDoS attacks. They tested using three different datasets and applied classical probability distributions to compute the probability from these datasets. For each unique IP address, they

computed individual probability values between 0 to 1. Then, compute entropy for each probability value and sum all entropy values within a time window for total entropy. Using this system, they have concluded that Information Entropy and Information distance provides better results in detecting both attack types when the order of generalized entropy is increased.

A paper by M. Aiello [133] presented a novel detection method in which analyzes specific spectral features of traffic over small time horizons. They focus on extrapolating proper spectral features from network traffic and study under which conditions legitimate and anomalous frequency spectra are significantly different. Those differences are derived over time horizons whose order of magnitude is 20s, thus leading to fast detection times. Those extrapolated data refer to real traffic traces, elaborated over the Local Area Network of their Institute, which uses Apache2 web server. Their analysis use statistical properties of the resulting signal and focuses on the quantity of data directed from the transport layer to the application layer. Even though analysis of data from the Application Layer is made, payload inspection of packets is not needed. To present their results, they used two metrics, one base on a simple average cover in their work and the other one used was Mutual Information. The authors justify that the originality of their approach relies on the choice of the traffic feature to infer the presence of DoS attacks. This feature was chosen because it drives an anomaly-based analysis of web traffic. Anomaly-based detection may be more adaptive than other tools derived from Machine Learning, as it simply looks at sudden changes of flows statistics, while providing good detection rates.

SeVen defense for Application Layer DDoS attacks based on the Adaptive Selective Verification (ASV) defense used for mitigating Network Layer DDoS attacks. This defense mechanism is proposed by Yuri G. Dantas et al. [134] and was formalized in the computational system *Maude* and demonstrated by using the statistical model checker *PVeStA* that can be used to prevent Application Layer DDoS attacks. ASV was designed for mitigating Network Layer DDoS attacks but is not enough for mitigating Application Layer DDoS attacks, as the protocols used by these attacks - such as HTTP - have a notion of state. SeVen thus extends ASV by incorporating into the defense a notion of state to work against Application Layer attacks. The validation of the project is made by simulation using the statistical model checker tool *PVeStA*, comparing results with a defense similar to used in the literature based on Traffic Analysis. They concluded that SeVen is both more robust and also leads to better traffic patterns than defense based on Traffic Analysis.

A method proposed by Arindom Ain et al [21], states that Rank Correlation measures such as Spearman Rank Correlation and Partial Rack Correlation (PRC), can quantify significant differences between attack traffic and legitimate traffic. The Spearman Rank Correlation coefficient (SRC) measures the strength of association between two random variables better and the Partial rank correlation computes correlation between two number of traffic instances within a sample. These Correlation coefficient measures are used to identify linear relationship between malicious and legitimate traffic. The authors tested using real-life datasets and made a comparison between the methods. Their final results were that PRC is more effective than SRC in differentiating malicious from legitimate traffic.

In the work of Idhammad et al. [84], they proposed a Detection System where they target HTTP DDoS attacks in a Cloud environment. Their method is based on Information Theoretic Entropy and Random Forest ensemble learning algorithm. It consists of three main steps: (I) Entropy Estimation, (II) Preprocessing, and (III) Network Traffic Classification. A time-based

sliding window algorithm is used to estimate the entropy of the network header features of the incoming network traffic. When the average of the features' entropy exceeds its normal range the preprocessing algorithm cleans and normalizes the network traffic data of the current time window. The preprocessed network traffic data is then classified into normal and HTTP DDoS traffic. Their tests were conducted using the CIDDS-001 dataset [135], first the proposed algorithm was tested on the dataset to check the compliance with the information given by the authors of the dataset. They were successful in see that in the first six hours of the test, were only detected normal traffic. The information is consistent with the information given by the dataset authors. Second, they tested five different Machine Learning Classifiers. They obtained a result were Random Forest was the most accurate and as a result chosen for the last test. The third and final test were made by using the whole proposed approach with the Time-based Sliding Window algorithm including the Random Forest Classifier. To validate their experiment, their work was compared with other state-of-the-art HTTP DDoS Detection methods in the literature. The entire proposed system achieved high detection performances compared to single classifiers tested directly on the dataset and to the state-of-the-art HTTP DDoS detection methods mentioned in their paper.

2.6 Conclusions

In this Chapter, a State-of-the-art study about Cloud Computing and its enabled technologies is done. As well as an an analysis of the Cloud Security Concerns, its relation with Denial of Service attacks and its types and operation methods, focusing on the Low-Rate Distributed Denial of Service attack variant.

Related researches about mitigation methods against Low-Rate DDoS are summarized and based on these studies, there are some matters that should be highlighted. These matters may be specified as both, consistent usability and ongoing issues:

1. Analyzing the Related Research 2.5 articles, one can visualize an evolution of the methods used throughout the years, rather be checking router flow or using Information Theory and Mathematical Methods. This means that different approaches may be used and studied in order to better detect and reduce Low-Rate Distributed Denial of Service attacks.

2. Despite there are some Information Theory-based DDoS attack detection methods for the Application Layer, which differ from techniques used in Classic defense mechanisms against DDoS attacks, they may have a deficiency in early detecting Low-Rate DDoS.

3. The majority of the methods presented in the Related Researches Section, focused on packet level detection while there are only few targeting flow level detection techniques. Those few Flow-based researches are mostly addressing the problem of volumetric DDoS attacks, even in the Application Layer.

4. Packet inspection methods may be very onerous from the hardware perspective, as it analyze every packet that passes through the network cards of the servers. But also, this method is proven to be more effective in early detection than network flow analysis.

5. The datasets presented in the experiments made by almost all of the works cited above contain or simulate real traffic. They are largely used throughout tests of traffic nature,

due to the difficulty of assemble physical equipment to simulate large environments such as Data Centers or Cloud Structures. But most of them are old. Real time simulation tests are not very common to find in the literature.

Chapter 3

Testbed Implementation and Results

3.1 Introduction

As the company Radware explained in [85], in order to successfully detect and mitigate Application-layer DDoS attacks such as HTTP and HTTPS floods or low and slow attacks, organizations should consider deploying on-site mitigation systems. Mitigation systems deployed in such proximity to the applications are designed to protect and can be detailed configurations to have a greater awareness to changes in network traffic flows in and out of the application servers, and therefore have a greater chance of detecting suspicious traffic on the application layer.

However, since one of the objectives of the method created is to detect Low-Rate Distributed Denial of Service attack before it reaches the target service, to have such a defense system operating near other services, may be concerning due to the amount of traffic a DDoS attack may generate and the speed of the traffic propagation. Therefore,

In this Chapter, an experimental testbed is presented in order to test the efficiency of the proposed system as well as the tests and results made against this system. The Section 3.2, clarifies the rules, technologies used, the victim and attacker architecture for the simulated environment infrastructure. The architecture of the method is illustrated based on traditional programming and defense techniques as gradually explained in Section 3.3. The tests and results are displayed in section 3.4, where the three test implementations are explained. In section 3.5 the results are analyzed and finally, in section 3.6, practical issues to perform the perform the tests are discussed.

3.2 Simulated Environment

The main idea is to test the Low-Rate DDoS attacks against the method created running inside a simulated infrastructure in real-time. This infrastructure should represent a Cloud environment with Load Balance and High Availability, as well as Virtual Machines with Apache web server, chosen to be the web server running on the system and receive the attack, as well as Minishift platform, to simulate a Platform-as-a-Service Environment.

3.2.1 Ground Rules and Limits

First, it was necessary to develop ground rules and limits, in order to establish a scope to test the attack and the functionality of the Hybrid Application in real time. These rules were:

1. The necessary time for the attacks to affect the proposed Cloud System, making it slow or inaccessible. A graph is created by the SlowHTTPtest tool after the tests in order to

compare the different time space with the amount of connections generated when the attack is running. This is displayed throughout the Section 3.4. This may be important to know for example, the time a system administrator has to intervene in case an attack is not contained before it reaches the target services or, in case the attack is insufficient to drive the service unavailable, so the system administrator may have time to examine the LDDoS attack as it unfolds in real time.

2. To inspect the effectiveness of the Hybrid Application, aside of detecting the Low-Rate DDoS attack, false positives were also analyzed within the log files.

3. Full control over the technologies used to build the environment and execute the tests.

After the specification of the rules, the Infrastructure is determined. The final design is displayed in the picture 3.1. Also, the technologies used for the implementation as well as the tests, were also specified.

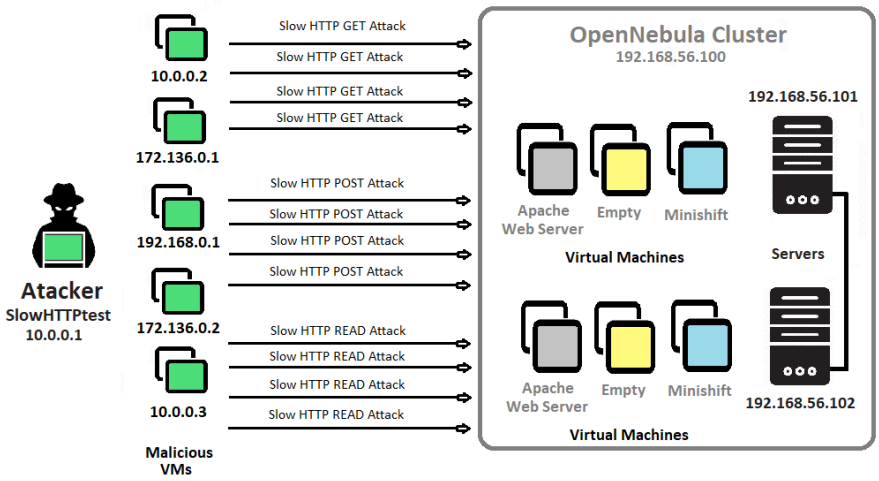

Figure 3.1: Schematic representation of the testbed implementation

3.2.2 Technologies Used

The specification of the technologies used in the project are presented. These technologies were chosen based on the proximity with a real world environment as possible. The idea is both, to determine the force of the attack and the efficiency of the method proposed. These technologies were:

- **OpenNebula**: It is an open source toolkit which allows users to transform existing infrastructure into an IaaS or PaaS Cloud with Cloud-like interfaces. The Sandbox Project is a virtual machine image with a pre-configured OpenNebula 5.6 front-end, a virtualization host using QEMU ready to execute virtual machines, and prepared images to offer a complete and rich Cloud experience. The idea was that users could be enrolled to build small-scale cloud infrastructures [136]. The scheme is illustrated in 3.4. OpenNebula uses port 9869, which the attack tool can be adjusted to aim specifically at this port.

- **Minishift**: Is a tool that helps one run OKD (OpenShift Platform-as-a-Service) locally by launching a single-node OKD cluster inside a virtual machine. With Minishift one can test

OKD or develop with it, day-to-day, on a local machine. Built around a core of Docker container packaging and Kubernetes container cluster management, OKD provides source container application platform [137]. Minishift uses port 8443, the attack tool can be adjusted to aim specifically at this port.

- **SlowHTTPtest**: It is an Application-Layer attack tool that drain concurrent connections pool by causing significant memory and CPU usage on the server [128]. This tool was chosen because it can perform the three variants of LDDoS explained in section 2.4.4. The command line used to launch the attacks are explained in the first test 3.4.1.

- **TShark**: Is a network protocol analyzer. It permits to capture packet data from a live network, or read packets from a previously saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file [138]. TShark is used to capture traffic in real time.

To fulfill the objective of the method regarding the performance of being efficient and lightweight, the selection of the technology was a combination between the environment implementation and the kinds of tests which were chosen.

3.2.3 Victim Infrastructure Settings

For the test environment, a cluster was built inside two Machines. Configured with Load Balancing and High Availability as displayed in 3.1. The infrastructure is described as follow:

- **OpenNebula Sandbox**: Two physical machines with OpenNebula Sandbox configured with Load Balancing and High Availability was the base system to perform the tests.

- **Apache Web Server Virtual Machines**: Two Web Server virtual machines were configured with default configurations were implemented in both machines to benefit from the Load Balancing and High Availability capability from the cluster.

- **Minishift Virtual Machines**: Two virtual machines were configured in order to analyze the impacts Application-Layer LDDoS attacks has inside a specific Platform-as-a-Service environment. Minishift was built as a single node container cluster and for this project, was built inside the OpenNebula Cluster.

3.2.4 Attacker Computer and Tools

The attacker computer used to perform the experiments was a simple personal computer and five Virtual Machines were configured to perform the attacks as presented in the picture 3.1. The choice was made specifically to display the disruptive power of these attacks using minimal efforts, against Cloud Services inside Platform-as-a-Service structures and also, against a Web Server structure. The test tool used to perform the Application-Layer LDDoS attacks was SlowHTTPtest. The configuration for each type of Low-Rate DDoS attack are specified later in 3.4.1.

3.3 Proposed Solution Details

The solution proposed to mitigate Low-Rate denial of service attacks is to create an Hybrid Application that increments the power of the Tshark tool combined with the patterns which this attack generates while sending its packets. Comparison specifications from Apache configuration file against these patters, can lead to distinguish malicious traffic from legitimate traffic.
First, the problems with the Apache modules when facing LDDoS are analyzed. Then, a thorough explanation about the stages of the Hybrid Application is presented.

3.3.1 Apache Modules Paradigm

As specified in subsection 2.3.7, Apache web server has models which may fight advanced DDoS threats. The `mod_reqtimeout` is one of these modules. An example of configuration using the `mod_reqtimout` is presented below:

```
<ifModule mod_reqtimeout.c>
   RequestReadTimeout header=25-50, MinRate=500, body=25
<IfModule>
```

In the `mod_reqtimeout` configuration above, one can see that the general idea is to differentiate these packets and eliminate the ones which satisfy this configuration.

The `header` data must be sent completely until 25 seconds. The client must send at a 500 bytes per second rate and the server will allow a maximum 50 seconds for the conclusion of the `header`. Also, the configuration let the `body` of the data to be sent entirely until 25 seconds. The problem is, the module permit the configuration of static limits in order to try to differentiate packets.

Using an attack tool or scripts to launch a Low-Rate Distributed Denial of Service attack, the perpetrator can easily modify the structure of the code to send a few packets, only to inspect if these packets will be eliminated or accepted by the server. Based on the definitions of `req_timeout`, the server will respond accordingly. Therefore, the attacker may be able to manipulate the code or script to accomplish its goals based on that response. For example, using SlowHTTPtest, the attacker can change a value for the interval time that the next data will be sent, change the connection rate or even the size of the `Content-Length`. In case the packet is rejected by the server, the attacker can fit the values properly until the server accept the packets. Therefore, the attacker may use these configuration values in the botnet computers to achive a Low-Rate Distributed Denial of Service attack.

Another problem with this static configurations is that if a legitimate client has a poor connection speed, it can be confused with a malicious traffic, and becoming unable to access the services. The `mod_qos` works in a similar way, parsing pre-established values in order to try to defend Apache web server.

Since the main goal for the Hybrid Application to work properly is to distinguish malicious traffic from legitimate traffic before these requests reach the target service, the Attack Pattern Combination method is used combined with the static information such as the ones the Apache modules use, but modified. After that, the information is logged, the packet considered with high probability of being malicious is dumped and the packet considered legitimate is sent normally to the destination service. The stages of the Hybrid Application are explained in the next subsections.

3.3.2 Packet Capture and Information Extraction with TShark

The Hybrid Application uses `TShark` to capture the packets and also, display only the specific features of the packets related with Low-Rate DDoS attacks, that have different patterns from legitimate packets. These information are used in the next state of the application. The patterns from each type of Low-Rate DDoS attack are cited in subsection 2.4.4 and are explained technically below:

- **Message 400, Bad request**: during the Slow HTTP GET attack (Slowloris) attack execution, the server generates massive number of `400: Bad Request` Errors.

- **Window Size with a constant size**: Each data sent advertises a window size. It is basically to tell the server about the size of the packet the server must expect. The size of the data content the client will send and the window size, are necessary for the server to receive that data. In legitimate traffic behavior, this number was observed to be more random for each IP address compared to the malicious traffic behavior, which sends regular window size for the same IP address connection made.

- **Suddenly Burst of connections**: The suddenly increase in the quantity of connections from the same IP addresses which are open during the attack in a small time frame. A specific characteristic of Denial of Service attacks, even for the Low-Rate types.

- **Large Content length**: For each data sent, the size of the `Content-Length` is specified. For the Slow HTTP POST attacks specifically, the size defined in the attack is larger than the size of the payload sent.

If some of those patterns are formed, the next stage of the hybrid Application will combine them with the quantity of the same source IP addresses. The TShark command with its respective filters to capture these specific patterns are:

```
tshark -r file.pcap (-i interface) -T fields -e frame.number -e ip.src
       -e ip.dst -e frame.len -e tcp.window_size -e http.response.code
       -e http.content_length -E header=y -E separator=;
```

Explaining the code above according to the Wireshark Official page [139]:

-r or -i : Select from a pcap file or choose which interface to capture.
-T : Set the format of the output when viewing decoded packet data, in this case the output was ”fields”.
-e : Specifies a field to display.
-> *frame.number* : Frame number set by the tool in the order of capture.
-> ip.src : Source IP address.
-> ip.dst : Destination IP address.
-> frame.len : Reports the total number of bytes that were transmitted bidirectionally in all the packets
-> window_size : Size of the window advertised by the client.
-> http.response.code : HTTP status-code generated based on the server understanding.
-> http.content_length: HTTP Content-Length Size advertised by the client.
-E : Set an option controlling the printing of fields when -T fields is selected.
-> header : Put a header with the names of the fields. The =n means no header defined.
-> separator : Set the separator character to use for fields. The semicolon (;) was chosen to separate the fields.

3.3.3 Attacks Pattern Combinations

After the patterns are discovered, they are stored inside a structure in the code. Then, these patterns are combined to identify the packets as malicious or legitimate. The algorithm for comparison is displayed and explained below:

Algorithm 1: Function to Compare Parameters extract from TShark

1 Function to Compare Parameters (a, b);
2 $PacketInfo * x \leftarrow (PacketInfo*)a$ ▷ $x \rightarrow extract\ \ parameter, a \rightarrow delegate\ \ parameter$;
3 $PacketInfo * y \leftarrow (PacketInfo*)b$ ▷ $y \rightarrow extract\ \ parameter, b \rightarrow delegate\ \ parameter$;
4 **if** $!(x \rightarrow parameter1)$ && $!(y \rightarrow parameter1)$ **then**
5 $\quad return\ \ 0$
6 **else if** $(!(x \rightarrow parameter1))$ **then**
7 $\quad return\ \ 1$
8 **else if** $(!(y \rightarrow parameter1))$ **then**
9 $\quad return\ \ -1$
10 $return\ \ (!strcmp\ \ (x \rightarrow parameter1,\ \ y \rightarrow parameter1))$

This Comparison function is used to inspect for repeated parameters. It compares the first parameter received with the next and each time this parameter appear, it increases the count value. For example, in order to compare Source IP addresses, the `parameter1` value inside the function can be replaced by `SourceIp` from the PacketInfo Structure.

This function may be considered generic since the comparison of the other parameters are executed using this same code. Inside the full code, this algorithm repeats itself for every necessary parameter besides the Source IP address.

After analyzing the quantity of repeated parameters, combinations are done as explained below:

- If the number of connections from the same `ip.src` combined with equal values of `windows_size`

and the `count` variable reaches 10% of `MaxRequestWorkers` (or for old Apache versions `MaxClients`) option obtained from the Apache configuration file, consider this packet as a possible threat.

- If the number of connections from the same `ip.src` combined with repetitive values of `http.content_length` and the `count` variable reaches 10% of `MaxRequestWorkers` (or for old Apache versions `MaxClients`) option retrieved from the Apache configuration file, consider this packet as a possible threat.
- If the number of connections from the same `ip.src` combines with repetitive values of `http.response.code` as `400: Bad Request` and the `count` variable reaches 10% of `MaxRequestWorkers` (or for old Apache versions `MaxClients`) option from the Apache configuration file, consider this packet as a possible threat.
- If the number of connections from the same `ip.src` reaches 10% of `MaxRequestWorkers` (or for old Apache versions `MaxClients`) option from the Apache configuration file and combines with repetitive values of `windows_size` and the `frame.len` is 10x bigger than the `window_size`, consider this packet as a possible threat.
- If the number of connections from the same `ip.src` reaches 10% of `MaxRequestWorkers` (or for old Apache versions `MaxClients`) option from the Apache configuration file and combines with repetitive values of `Content_Length` and the `frame.len` is 10x smaller than the `window_size`, consider this packet as a possible threat.

The value of 10% was chosen based on the size of the Cloud infrastructure created for this work and after the first test which evaluated the potential of the attack tool.

3.3.4 Packet Classification and Catalog

After the comparison, the IP addresses considered malicious are logged in a specific file for further analysis. Tags are added to specify the reason the IP addresses were dumped, they are divided in:

- Suspicious Content, Possible Slow GET
- Suspicious Content, Possible Slow POST
- Suspicious Content, Possible Slow Read

Packets with suspicious content are eliminated and packets with no suspicious behavior are accepted.

3.4 Test Implementation

Before the main tests, preliminaries tests were executed in order to analyze the attacker computer and victim infrastructure architecture. This analysis were necessary to determine:

1. If the computer used as the perpetrator would be effective in attacking the OpenNebula infrastructure.
2. The necessary amount of virtual machines to affect the services inside the infrastructure.

3. Different tests were executed directly to the OpenNebula port 9869, Minishift port 8443 and HTTP protocol port 80.

After these preliminaries tests, the conclusions were:

1. When using more than five virtual machines, similar results were achieved.
2. The attack done specifically against the Minishift and the Apache web server were also similar with the attack performed against the OpenNebula infrastructure. The difference is the proportion since the virtual machines for these services are configured inside the OpenNebula Cloud.

After these conclusions, the final decision was that the attack tests should be executed directly against the OpenNebula infrastructure. This decision became clear after observations of the behavior when the attacks performed specifically against the services inside the infrastructure had similar results when the same attacks were performed against the OpenNebula itself. The main difference is that when the attack is performed against the whole infrastructure, the Minishift and Apache were also affected. Therefore, it is important to express that the behavior explained in the tests and results below, are from the infrastructure as a whole. The detailed test implementation is explained in this Section. Three tests were executed and its parameters and specifications are presented.

3.4.1 Test 1: SlowHTTPtest Tool Effectiveness and Adaptations

For this first test, five different virtual machines were used to attack the OpenNebula Cluster simultaneously. The chosen tool to perform the attack was the SlowHTTPtest [128]. The advantage of using this tool, is that the three types of Application-Layer Low-Rate DDoS can be performed. As aforementioned in section 3.2, a graph is created after each one of the attack tests are finished. This graph evaluates the number of connections and time necessary to affect the infrastructure. Despite of already being confirmed to be used to disrupt services from major companies as stated in [14, 16, 17, 15], the reason for this first test is to measure proportionality. The idea is to discover which configuration is necessary in the attack tool to start disrupt the OpenNebula Cloud System services. Therefore, the ultimate goal of this first test is to evaluate the configuration of the tool when the attacks start to be effective against the Cloud System, and retrieve the configurations for the upcoming tests.

Multiple tests for each type of attack were developed in order to achieve the ideal configuration against the Cloud environment created. The tests are described as: A) Slow HTTP GET test, B) Slow HTTP POST test and C) Slow Read test. In the Section 3.5, the table 3.1 compares the number of connections required to be configured in the attack tool against the availability of the Cloud environment.

Finally, the configuration required for each Low-Rate DDoS attack type start to interfere with the availability of the Cloud environment is explained below:

A) Slow HTTP GET test

The fist attack performed was the Slow HTTP GET (Slowloris) attack [125, 122] and the configuration used is described as follow:

```
slowhttptest -c 4000 -H -g -o slowloris -i 10 -r 200 -t GET -u
https://192.168.56.101/index.php -x 24 -p 3
```

For this attack, the below configurations was used.

```
-c : Number of Connections.
-H : Specify to slow down in headers section.
-g : Generate statistics in CSV and HTML formats.
-o : Custom output file path and/or name.
-i : Interval between follow up data in seconds, per connection.
-r : Connection rate.
-t : Custom verb to use.
-u : Target URL, the same format used in the browser.
-x : Max length of follow up data.
-p : Timeout to wait for HTTP response on probe connection.
```

B) Slow HTTP POST test

The second attack performed was the Slow HTTP POST Attack (R.U.D.Y). The code used to execute the attack is presented bellow:

```
slowhttptest -c 1000 -B -g -o slowpost -i 110 -r 200 -s 8192 -t POST -u
http://192.168.56.101/form.php -x 10 -p 3
```

For this attack was used the below configurations.

```
-c : Number of Connections.
-B : Specify to slow down in message body.
-g : Generate statistics in CSV and HTML formats.
-o : Custom output file path and/or name.
-i : Interval between follow up data in seconds, per connection.
-r : Connection rate.
-s : Value of Content-Length header, if -B specified.
-t : Custom verb to use.
-u : Target URL, the same format used in the browser.
-x : Max length of follow up data.
-p : Timeout to wait for HTTP response on probe connection.
```

C) Slow HTTP Read test

The third attack was the Slow HTTP Read attack. The code is displayed and explained below:

```
slowhttptest -c 4000 -X -r 1000 -g -o slowread -w 10 -y 20 -n 5 -z 32
-u http://172.168.56.101 -p 5 -l 350
```

Code explanation below:

```
-c : Number of Connections.
-X : Enables slow read test.
-r : Connection rate.
-g : Generate statistics in CSV and HTML formats.
-o : Custom output file path and/or name.
-w : Start of range the advertised window size would be picked from.
-y : End of range the advertised window size would be picked from.
-n : Interval between read operations from receive buffer.
-z : Bytes to read from receive buffer with single read() operation.
-u : Target URL, the same format used in the browser.
-p : Timeout to wait for HTTP response on probe connection.
-l : Test duration in seconds.
```

3.4.2 Test 2: SlowHTTPTest vs Hybrid Application in real-time

This test was conducted using the SlowHTTPTest tool against the Cloud environment created and the Hybrid Application running inside the the OpenNebula Cloud environment. The objective was to analyze the effectiveness of the Hybrid Application in real-time against malicious attacks and also, the performance of the Opennebula as well as the Apache web service and the Minishift Platform-as-a-Service inside the Cloud environment. To perform this test, some standards were adopted:

1. the same configuration of the SlowHTTPTool aforementioned in Section 3.4.1 were used.
2. The attacks were conducted with a specific amount of time.

3.4.3 Test 3: Hybrid Application Differentiation Capability

This test was performed using an existent dataset called WRCCDC dataset. This dataset is a combination of legitimate traffic and attack traffic. The Hybrid Application was modified to receive the file and analyze the packets captured. The test was conducted based on the combination details explained in Section 3.3.3.

This dataset is a collection of traffic extracted from the Western Region Collegiate Cyber Defense Competition. The WRCCDC main goal is that students assume administrative and protective duties of an existing commercial network - typically a small company with about 50 users, 7 to 10 servers, and common Internet services such as a web server, a mail server, and an e-commerce site. The student teams are scored on their ability to detect and respond to outside threats, maintain availability of existing services such as mail servers and web servers, respond

to business requests such as the addition or removal of additional services, and balance security needs against business needs [12].

3.5 Results Analysis

The results of each test are explained based on the different types of Low-Rate DDoS attack.

3.5.1 Results of Test 1

The first test was conducted using the configurations aforementioned in Section 3.4.1 for the Slow HTTP GET attack variant, and the results are described as follow:

A) Slow HTTP GET attack Results

Based on the information gathered, the results are:

1. The Slowloris attack was effective against the Cluster when used more than 4000 connections in the five Virtual Machines used for the attack. After 300 connections, the Minishift and the Apache web server started to be affected. The rest of the connections that were kept in the queue also affected the server later.

2. Since the OpenNebula Cluster can handle larger numbers of connections, the connections from the Slowloris attack did not reach the peak. As the graph 3.2 presented, less than 3000 connections were necessary to disturb the Cloud Environment.

3. Within 7 seconds, the services started to be affected and became slow. When the attack opened close to 500 connections, the services stopped and stayed down for around 45 seconds.

4. After, the services were slow and oscillated for about 20 seconds, the rest of the connections, which were pending, started to connect to the services and when they reached about 2000 connections, the services stopped again for about 20 more seconds.

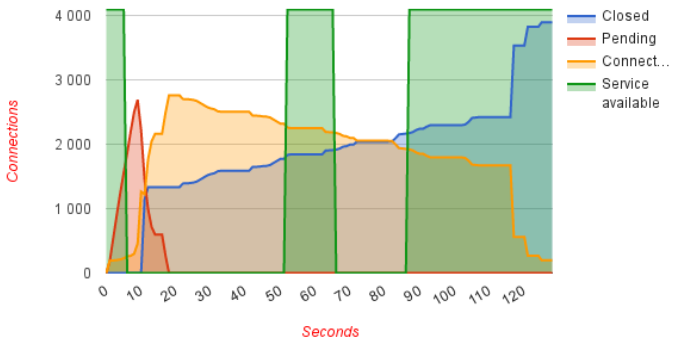

Figure 3.2: Slow HTTP GET (Slowloris) Attack test Graph

This type of attack is considered ineffective against Clusters because of the Load Balancing feature. However, this test proved to be different. Even with these features configured inside the Cloud environment, the Slow HTTP GET confirmed to be a success in affect this Infrastructure.

The results can be seen in graph 3.2.

B) Slow HTTP POST attack Results

Based on the information collected the results were:

1. The Slow POST attack results may be similar to the results presented by the Slowloris attack but based on the tests displayed, it demonstrated to be more disruptive for the OpenNebula Cloud infrastructure.
2. The test tool was configured with 1000 maximum connections, as the graph 3.3 illustrates, and after that achieved about 300 connections, the server stopped.
3. The services stopped for about 25 seconds, and went back online for just a few seconds, then it stopped again when approximately 400 attack connections were established.
4. The services went online again for about 25 seconds before they stop again for about 16 seconds.
5. The time the services inside the Cluster were online, they were oscillating. Therefore, not functioning in.

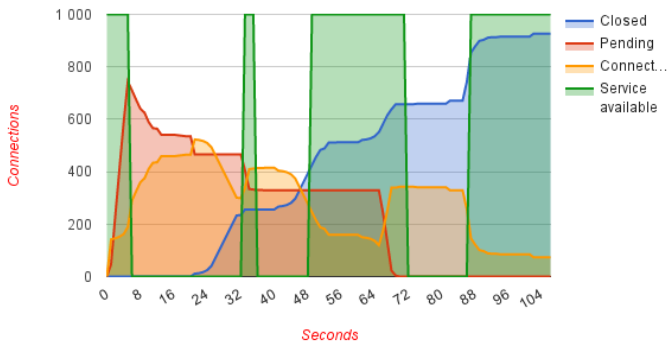

Figure 3.3: Slow HTTP POST Attack test graph

The Slow HTTP POST attack graph can be seen in 3.3. Similar to Slow HTTP GET and also confirming one of the characteristics of Low-Rate DDoS explained in 2.4, this attack type works in pulses, bringing the services online and offline on a time variation. However, different from Slow HTTP GET, Slow HTTP POST was able to bring the services down using less connections. The services stayed down for longer than the other types of attack studied in this Section.

C) Slow HTTP Read attack Results

Based on the information collected, the results are displayed as follow:

1. Slow Read attack was not effective against this infrastructure compared to the other types of attacks presented in this Section.
2. For this attack, it was used 4000 connections and these connection, achieved the peak of connections inside the services, Slow Read attack was able to bring the service down after about 15 seconds.

3. The services recovered after about 8 seconds, the Slow Read attack was unable to affect the services again. Even with the 4000 connections still running inside the Cluster.

As the graph 3.4 presents, this attack was able to affect the server but the services only stopped for approximately 8 seconds. This time may still be considered effective to combine with other attacks but compared with Slow HTTP GET and Slow HTTP POST, Slow Read attack was not as effective as the other types of Low-Rate Attacks, against the Cloud Infrastructure created.

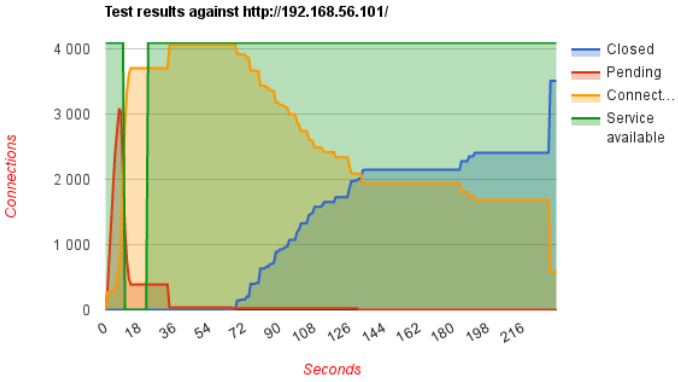

Figure 3.4: Slow HTTP READ Attack test Graph

A comparison of the test parameters can be seen following the picture 3.5. These parameters are explained in detail by the SlowHTTPtest Tool aforementioned in 3.4.1, 3.4.2 and 3.4.3 respectively.

Test parameters	
Test type	SLOW HEADERS
Number of connections	4090
Verb	GET
Content-Length header value	4096
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	240 seconds
Using proxy	no proxy

Test parameters	
Test type	SLOW BODY
Number of connections	1000
Verb	POST
Content-Length header value	8192
Extra data max length	22
Interval between follow up data	110 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	240 seconds
Using proxy	no proxy

Test parameters	
Test type	SLOW READ
Number of connections	4090
Receive window range	10 - 20
Pipeline factor	1
Read rate from receive buffer	32 bytes / 5 sec
Connections per seconds	1000
Timeout for probe connection	5
Target test duration	350 seconds
Using proxy	no proxy

Figure 3.5: Comparison between Parameters of Low-Rate Distributed Attacks Variations

The table 3.1 was generated to compare the different connection values necessary for each attack type when they start to affect the Cloud environment.

Connections	SlowGET	SlowPOST	SlowRead
100	up	up	up
200	up	up	up
300	slow	down	up
400	slow	down	up
500	down	down	up
1000	down	down	up
2000	down	-	up
3000	down	-	down
4000	-	-	down
5000	-	-	-

Table 3.1: Number of Connections Necessary to Affect the Cloud Environment

3.5.2 Results of Test 2

The results of the test with the Hybrid Application in Real-time against the three types of Low-Rate DDoS are displayed in table 3.2, table 3.3 and table 3.4. Each table represents Slow HTTP GET, Slow HTTP POST, Slow HTTP Read, respectively. The tables are divided by the time in seconds, of which four instances of 20 seconds each are analyzed and the results are displayed as follow:

Time	Connections	Received Packets	Dropped
0 - 20 seconds	2,327	10,244	9,743
20 - 40 seconds	3,234	7,707	7,465
40 - 60 seconds	3,287	3,105	2,976
60 - 80 seconds	2,937	2,398	2,312

Table 3.2: Slow HTTP GET attack vs Hybrid Application in Real-Time

Time	Connections	Received Packets	Dropped
0 - 20 seconds	1000	4125	4081
20 - 40 seconds	419	2015	1993
40 - 60 seconds	150	1350	1342
60 - 80 seconds	-	-	-

Table 3.3: Slow HTTP POST attack vs Hybrid Application in Real-Time

Time	Connections	Received Packets	Dropped
0 - 20 seconds	658	4,621	4,543
20 - 40 seconds	1,563	5,534	5,441
40 - 60 seconds	2,389	5,473	5,348
60 - 80 seconds	3,200	6,563	6,388

Table 3.4: Slow HTTP Read attack vs Hybrid Application in Real-Time

With the conclusion of these tests, some points deserve to be mentioned:

1. The results presented are from the attack tool configured to attack specifically the OpenNebula infrastructure. Tests were performed against the specific port of OpenNebula but the results were very similar.

2. The Hybrid Application was able to drop about 96% of the incoming traffic from the Slow HTTP GET attack.

3. The Hybrid Application was able to drop about 99% of the incoming traffic from the Slow HTTP POST attack.

4. The Hybrid Application was able to drop about 98% of the incoming traffic from the Slow HTTP GET attack.

5. Services were not affected during the performance of these tests. All the services were online the full time.

3.5.3 Results of Test 3

The objective of this test is to analyze the capability of the Hybrid Application. For this, the dataset from the Western Regional Collegiate Cyber Defense Competition [12] was used.

The results for this test presented that the application was very useful in detecting and separating the packets. As well as eliminating the packets considered malicious. Analyzing the log files and comparing with the dataset, some IP addresses were considered being malicious but they should not. 6% of these packets considered malicious were false positives. The table 3.5 demonstrates the numbers divided in 200.000 IP captured per specific time.

Packets	Malicious	Legitimate	Time
200,000	0	200,000	1 minutes
200,000	3	199,997	1 minutes
200,000	545	199,455	2 minutes
200,000	10624	189,376	7 minutes
200,000	288	199,712	1 minute

Table 3.5: Malicious and Legitimate Traffic Capture by the Hybrid Application Tool

Based on the information gathered, the results are displayed:

1. About 1,000,000 packets were captured.
2. The Hybrid Application took about 12 minutes to finish.
3. approximately 1400 packets were collected and analyzes per second.
4. Analyzing the Dataset and the log files, 6% of the packets were considered False-Positives.

The results of the hybrid Capability displayed the possibility of differentiate the packets based on observe combining patterns of the types of Low-Rate Distributed Denial of Service attack. The application presented efficiency in capture the packets and combine the patterns.

3.6 Practical Issues

The tests were performed using the tools and an Infrastructure was build in order to test the performance of the Hybrid Application. Based on that, some Practical Issues can be cited:

1. In tests which used Real-Time, the attacks were also conducted against the Apache web server and Minishift Platform-as-a-Service individually, but due to the hardware deficiency (limited memory and CPU) of the infrastructure, results could not be conclusive.
2. The final command line to generate the attacks with SlowHTTPtest was obtained from testing other configurations. Slowloris was ineffective when used less connections than the values specified in table 3.1.
3. The Apache Servers are using their default configuration. Nothing was changed to perform this tests.
4. The OpenNebula Sandbox is using its default configuration. Nothing was changed throughout the testbed implementation.

5. The Minishift Platform-as-a-Service is using its default configuration. Nothing was changed during the testbed implementation.

3.7 Conclusions

In this Chapter, the experimental testbed was discussed. The explanation about the technologies used and the specific features of the proposed solution were also presented.

The use of OpenNebula SandBox to simulate a real Cloud architecture, as well as Minishift used to simulate a Platform-as-a-Service Environment. Apache web server was used to represent the server instance inside a Cloud environment and two other virtual machines were created with different Linux distributions to complete the simulated environment. The specification of the SlowHTTPtest as the attack tool used to perform the three LDDoS variants against the Application-Layer studied in this work. This Chapter explained both, the victim infrastructure and the attack instances that were prepared to perform the tests described in Section 3.4.

This Chapter also specified the tests performed with the SlowHTTPtest tool against the Cloud Infrastructure created, as tests using a dataset which is a collection of traffic inside specific servers or network equipment, that can be used to test traffic related implementations. The results of these tests were also explained and demonstrated in graphs and tables throughout the Section.

The information obtained after performing the attacks presented that even working from only a single computer and five virtual machines, the Low-Rate DDoS attack can be effective. The Cloud Infrastructure was affected by the three types of Low-Rate DDoS as demonstrated in the results Section 3.5.

Chapter 4

Conclusions

4.1 Main Conclusions

The objective of this work was to study the Low-Rate Distributed Denial of Service threat and elaborate a plan to mitigate it against Cloud Services, specifically analyzing Platform-as-a-Service. The Low-Rate DDoS attack is on the rise and has been used by numerous hacker groups and communities to disrupt specific services as literature cite in [89, 19, 90, 80]. Throughout this work, the challenge considered was to identify this threat, since it works different from traditional Denial of Service attacks and elaborate a method to separate malicious traffic from legitimate traffic. The traditional Distributed Denial of Service attack defenses are discussed as not effective against the LDDoS due to its nature and more advanced defense techniques may be expensive and difficult to assemble.

Studies from other researches were done and presented that specific subjects on this threat is limited. some of methods proposed focus on packet analysis which proved to be a more efficient way to identify Low-Rate DDoS. Also, other methods proposed focuses on flow analysis, but there are also limited researches about it. Based on this, a Hybrid Application was proposed using TShark Packet analyzer to capture the packets and filter the needed information. A combination of events were assembled based on observance of the specific characteristics of the Low-Rate DDoS variants in real-time attacks performed numerous times against a Cloud environment built to emulate real-world infrastructure.

Two testbed scenarios were established and three tests were done. The first Scenario was created using OpenNebula Cloud Environment with High-Availability and Load Balancing with the objective to test the attack tool capability as well as the Hybrid Application in different manners. The second scenario idea was executed by first, modifying the Hybrid Application to capture HTTP packets from the 2018 Western Regional Collegiate Cyber Defense Competition dataset, in order to analyze the capability of the Hybrid Application to separate malicious traffic from legitimate traffic.

The first test was executed using the SlowHTTPTest tool directly attack the Cloud infrastructure created. Various attacks were performed and the final result was a specific combination of configuration patterns where the tool affects the Cloud infrastructure and also provides time for the observation of the peculiarities of the Low-Rate DDoS attack types. Based on the configuration results of the first test, a second test was executed. This test had the objective of analyze the performance of the Hybrid Application in detecting the Low-Rate DDoS variants against the Application Layer and based on its distinct characteristics. The results were very favorable in the detection rate of the Low-Rate threat types. The final test was execute using the WRCCDC dataset, in order to evaluate the efficiency of the Hybrid Application in separate legitimate traffic from malicious traffic. The results demonstrated that the method proposed presented effectiveness in detecting malicious traffic and the systems inside the Cloud infras-

tructure were not affected during this test. Therefore, the Hybrid Application built focused on specific patterns that the Low-Rate Distributed Denial of Service attack have, proved to be a lightweight and effective manner to analyze contents inside a packet in order to mitigate this threat.

4.2 Future Work

Some guidelines to better test the application and use the idea to defend against Low-Rate Distributed Denial of Service attacks and other threats:

- Test in a more realistic environment.
- Revise the Hybrid Application for better performance, specially with longer duration attacks.
- Improve the approach against the specific characteristics of the Low-Rate DDoS types.
- Test with a mixture of legitimate traffic and malicious traffic in real time.
- Creating a particular sniffer instead of using an existing one, achieving more flexibility.

References

[1] Louis Columbus, "2017 State Of Cloud Adoption And Security," 2017, Last accessed, October 2018. Found Online at https://www.forbes.com/sites/louiscolumbus/2017/04/23/2017-state-of-cloud-adoption-and-security/. xix, 1

[2] Edwin Schouten, "Cloud computing defined: Characteristics service levels," 2014, Last accessed, October 2018. Found Online at https://www.ibm.com/blogs/cloud-computing/2014/01/31/cloud-computing-defined-characteristics-service-levels/. xix, 11

[3] Iain Swanson et. al., "Virtual Environments Support Insider Security Violations," *Australian Digital Forensics Conference*, 2008. xix, 22

[4] Christos Douligeris & Aikaterini Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks. Vol. 44, no. 5, p. 643-666*, 2003. xix, 29, 31, 32, 34, 35, 36

[5] Akashdeep Bhardwaj et al., "DDoS Attacks, New DDoS Taxonomy and Mitigation Solutions – A Survey," *International conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 2016. xix, 31, 32

[6] IBM Global Education Group, "Virtualization in Education," *White Paper*, 2007. 1, 21, 22

[7] Sunny Tsiao, *Read you loud and clear! The story of NASA'S spaceflight tracking and data network*. U .S. Government Printing Office, 1970. 1, 13

[8] Li Da Xu et al., "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2233 - 2243*, 2014. 1

[9] Deloitte Consulting LLP team, "Everything-as-a-service: Modernising the core through a services lens," *Tech Trends 2017: The kinetic enterprise*, 2017. 1, 8

[10] Qi Zhang et al., "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications. Vol. 1, no. 1, pp. 7-18*, 2010. 1, 8, 9, 10, 26

[11] Charalampos Patrikakis et al, "Distributed Denial of Service Attacks," *The Internet Protocol Journal, Vol. 7, no. 4*, 2004. 2, 29

[12] Western Regional Collegiate Cyber Defense Competition, "WRCCDC Dataset," 2018, Last accessed, October 2018. Found Online at http://www.wrccdc.org/. 2, 63, 67

[13] Akamai Technologies, "Quaterly Security Reports," 2017. Last accessed, October 2018. Found Online at https://www.akamai.com/uk/en/about/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp. 2, 46

[14] Pierluigi Paganini, "OVH Hosting hit by 1Tbps DDoS Attack, The Largest one ever seen," 2016. Last accessed, October 2018. Found Online on http://securityaffairs.co/wordpress/51640/cyber-crime/tbps-ddos-attack.html. 2, 60

[15] Swati Khandelwal, “World’s Largest 1Tbps DDoS Attack launched from 152.000 hacked smart devices ,” 2016. Last accessed, October 2018. Found Online at https://thehackernews.com/2016/09/ddos-attack-iot.html. 2, 60

[16] Darrell Etherington and Kate Conger , “Large DDoS attacks cause outages at Twitter, Spotify, and other sites,” 2016. Last accessed, October 2018. Found Online at https://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/. 2, 60

[17] Conner Forrest , “Dyn DDoS attack: 5 takeaways on what we know and why it matters,” 2016. Last accessed, October 2018. Found Online at https://www.techrepublic.com/article/dyn-ddos-attack-5-takeaways-on-what-we-know-and-why-it-matters/. 2, 60

[18] Imperva Incapsula, “Global DDoS Threats Landscape,” Last accessed, October 2018. Found Online at https://www.incapsula.com/ddos-report/ddos-report-q4-2017.html. 2

[19] E. Cambiaso et al., “Slow DoS attacks: definition and categorisation,” *International Journal of Trust Management in Computing and Communications. Vol. 1, no. 3-4*, 2013. 3, 29, 69

[20] Túlio A. Pascoal et al., “Módulo de Proteção contra Ataques de Negação de Serviço na Camada de Aplicação: uma Análise de Qualidade de Serviço e Experiência de Usuário,” *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2017. 4, 40, 47

[21] Arindom Ain et al., “Rank Correlation for Low-Rate DDoS Attack Detection: An Empirical Evaluation,” *International Journal of Network Security, Vol. 18, pp. 474-480*, 2016. 4, 50

[22] Yang Xiang et al., “Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics,” *IEEE Transactions on Information Forensics and Security, Vol. 6, No. 2, pp. 426-437*, 2011. 4, 41, 42, 48, 49

[23] John W. Rittinghouse and James F. Ransome, *Cloud Computing: Implementation, Management and Security*. Taylor and Francis Group, LLC, 2010. 7, 15, 16, 21, 25, 26

[24] Shui Yu, “Distributed Denial of Service Attack and Defence,” 2013. Monograph. 7, 27

[25] Barrie Sosinsky, *Cloud Computing Bible*. Wiley Publishing, Inc, 2011. 7, 9, 10, 11, 12, 18, 23

[26] Peter Mell & Timothy Grance, “The NIST definition of Cloud Computing,” *NIST Special Publication 800-145*, 2011. 8, 9, 10

[27] Cloud Security Alliance, *Security Guidance for Critical Areas of Focus in Cloud version 3.0*. Cloud Security Alliance, 2011. 8, 9, 12

[28] Borko Furht & Armando Escalante, *Handbook of Cloud Computing*. Springer Science+Business Media, 2010. 8, 9, 12, 13, 14, 15

[29] Germán Goldszmidt & Indrajit Poddar, “Challenges and Architectural Patterns,” *Develop and Deploy Multi-Tenant Web-delivered Solutions using IBM middleware*, 2008. 8, 9

[30] Adrián Juan-Verdejo & Bholanathsingh Surajbali, "XaaS Multi-Cloud Marketplace Architecture Enacting the Industry 4.0 Concepts," *Doctoral Conference on Computing, Electrical and Industrial Systems (DoCEIS): Technological Innovation for Cyber-Physical Systems. Vol. 470, pp. 11-23*, 2016. 8

[31] Yucong Duan et. al, "Everything as a Service (XaaS) on the Cloud: Origins Current and Future Trends," *IEEE 8th International Conference on Cloud Computing*, 2016. 8

[32] Lizhe Wang et. al, "Cloud computing: A Perspective study," *New Generation Computing. Vol. 28, no. 2, pp. 137-14*, 2008. 8, 19, 21, 23

[33] Bhaskar Prasad Rimal et. al, "A taxonomy and Survey of Cloud Computing Systems," *Fith International Joint Conference on INC, IMS and IDC*, 2009. 9

[34] Kai Hwang et. al, *Distributed and Cloud Computing: From Parallel Processing to Internet of Things*. Elsevier INC publisher, 2014. 10, 12, 13, 20, 22, 23, 24

[35] Anthony T. Velte et. al, *Cloud Computing: A Practical Approach*. McGraw-Hill, 2010. 10, 11, 14, 16, 24, 29

[36] Sushil Bhardwaj et. al, "Cloud Computing: A study of Infrastructure as a Service (IaaS)," *International Journal of Engineering and Information Technology, Vol. 2, no. 1, pp. 61-63*, 2010. 10

[37] Rajkumar Buyya et. al, *Cloud Computing: Principles and Paradigms*. John Wiley and Sons, INC publications, 2011. 11, 12, 13, 15, 21, 25

[38] Wei Sun et. al, "Software as a Service: An Integration Perspective," *Proceedings of the Fifth International Conference. Vienna, Austria*, 2007. 12

[39] Yossi Gilad et al., "CDN-on-Demand: An Affordable DDoS Defense via Untrusted Clouds," *Network and Distributed System Security Symposium*, 2016. 12, 40, 41

[40] Akashdeep Bhardwaj et al., "Three Tier Network Architecture to Mitigate DDoS Attacks on Hybrid Cloud Environments," *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, 2016. 12, 37, 39, 41

[41] Claudio Mazzariello et al., "Integrating a Network IDS into an Open Source Cloud Computing Environment," *IEEE 6th International Conference on Information Assurance and Security*, 2010. 12, 36

[42] Mike Ebbers et. al, *Introduction to the New Mainframe, 3rd Edition*. IBM RedBooks, 2011. 13, 14, 15

[43] Jim Elliott, "Presentation: History and Evolution of IBM Mainframes," 2005. 13

[44] IBM Corporation, *z/OS Basic Skills Information Center: Mainframe concepts*. IBM Corporation, 2009. 13

[45] David Walden & Tom Van Vleck, "Compatible Time-Sharing System (1961-1973)," *Fiftieth Anniversary Commemorative Overview, IEEE Computer Society*, 2011. 13

[46] Fernando J. Corbató et. al., "An Experimental Time-Sharing System," *Proceedings of the Spring Joint Computer Conference. pp. 335-344*, 1962. 13

[47] Douglas E. Comer, *Internetworking with TCP/IP: Principles, protocols and architecture*, vol. 1. Prentice Hall, 6th ed., 2013. 13, 33, 34, 35, 36

[48] Tim Mather et. al., *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Media, Inc., 2009. 13, 27

[49] SUSE LLC and contributors , *Virtualization Guide*. SUSE LLC, 2016. 13, 14, 21, 24, 25

[50] Paul Barham et al., "Xen and the Art of Virtualization," *SOSP '03 Proceedings of the 19th ACM symposium on Operating systems principles, Pages 164-177* , 2003. 14, 21, 23

[51] Rich Uhlig et al., "Intel virtualization technology," *IEEE Computer Society, Vol.38, no. 5, pp. 48 - 56*, 2003. 14, 21, 24, 25

[52] Brian J. S. Chee and Curtis Franklin Jr., *Cloud Computing: Technologies and Strategies of the Ubiquitous Data Center*. CRC Press, Taylor and Francis Group, LLC, 2010. 14, 15

[53] Microsoft Corporation, "What is Virtualization," Last accessed, December 2017. Found online at https://azure.microsoft.com/en-us/overview/what-is-virtualization/. 14, 21, 22

[54] Dirk Krafzig et. al., *Enterprise SOA: Service-oriented Architecture Best Practices, 1st Edition*. Prentice Hall, 2004. 15, 16, 18

[55] Thomas Erl, *SOA: Principles of Service Design*. Prentice Hall, 2008. 15, 17, 18

[56] Michel S. Soares et al., "Characterization of the Application of Service-Oriented Design Principles in Practice: A Systematic Literature Review," *Journal of Software, Vol. 11. no. 4*, 2016. 18

[57] George Feuerlicht, "Enterprise SOA: What are the benefits and challenges?," *Systems Integration*, 2006. 18, 19

[58] Fakorede Oluwatoyin Johnson, "An Investigation into the Implementation Issues and Challenges of Service Oriented Architecture," *Master Thesis*, 2007. 19, 20, 21, 26

[59] Wei-Tek Tsai et. al., "Service-Oriented Cloud Computing Architecture," *Seventh International Conference on Information Technology*, 2010. 19

[60] Quan Z. Sheng et al., "Web services composition: A decade's overview," *Information Sciences. Vol. 280, pp. 218-238*, 2014. 19

[61] Steve Graham et. al., *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI. 2nd Edition*. Sams Publishing, 2004. 19

[62] World Wide Web Consortium, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) ," 2007. Last accessed, October 2018. Found Online at https://www.w3.org/TR/soap12-part1/. 20

[63] World Wide Web Consortium, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language) ," 2007. Last accessed, October 2018. Found Online at https://www.w3.org/TR/wsdl/. 20

[64] Organization for the Advancement of Structured Information Standards, "OASIS Standards," Last accessed, November 2018. Found Online at https://www.oasis-open.org/standards. 20

[65] Judith Hurwitz et al., *Service Oriented Architecture for Dummies*. Wiley Publishing, Inc., 2007. 20

[66] Adnan Masood, "Cyber Security for Service Oriented Architectures in a Web 2.0 World: An Overview of SOA Vulnerabilities in Financial Services," *IEEE International Conference on Technologies for Homeland Security*, 2013. 21

[67] Thomas Erl et al., *SOA with REST: Principles, Patterns Constraints for Building Enterprise Solutions with REST*. Prentice Hall, 2012. 21

[68] Yefim V. Natis, "Service-Oriented Architecture Scenario," *Gartner Publication, Inc*, 2003. 21

[69] Mohamed Al Morsy et. al., "An Analysis of the Cloud Computing Security Problem," *Computing Research Repository*, 2016. 21

[70] Namkyun Baik et. al., "Effective DDoS Attack Defense Scheme Using Web Service Performance Measurement," *Fourth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2012. 21

[71] Miguel G. Xavier et al., "Performance Evaluation of Container-based Virtualization for High Performance Computing Environments," *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 233-240*, 2013. 23

[72] Gabriel Cephas Obasuyi & Arif Sari, "Security Challenges of Virtualization Hypervisors in Virtualized Hardware Environment," *International Journal of Communications, Network and System Sciences, vol. 8, pp. 260-273*, 2015. 23

[73] Alison Holloway & Pieter Gordebeke, *Oracle VM User's Guide*. Oracle Corporation, 2011. 23

[74] Thomas Erl et al, *Cloud Computing: Concepts, Technology, Architecture. 1st edition*. Prentice Hall, 2013. 24, 25

[75] Anish Babu S. et al., "System Performance evaluation of Para-virtualization, Container virtualization and Full-virtualization using Xen, OpenVZ and XenServer," *Fourth International Conference on Advances in Computing and Communications*, 2014. 24

[76] Chris Greamo & Anup Ghosh, "Sandboxing and Virtualization: Modern Tools for Combating Malware," *IEEE Security Privacy, Vol. 9, no. 2, pp. 79-82*, 2011. 24

[77] Gustavo Alessandro Andrade Santana, *Data Center Virtualization Fundamentals*. Cisco Press, 2014. 25

[78] Diego Perez-Botero et al., "Characterizing Hypervisor Vulnerabilities on cloud computing," *Proceedings of the 2013 international workshop on Security in cloud computing, pp. 03-10*, 2013. 26

[79] Open Web Application Security Project, "OWASP 2017: The Ten Most Critical Web Application Security Risks," , 2017. 26, 38

[80] Cloud Security Alliance, "The Treacherous 12 - Top Threats to Cloud Computing Industry Insights," , 2017. 27, 29, 69

[81] Arbor Networks, Inc, "Insight into the Global Threat Landscape," *13th Worldwide Infrastructure Security Report*, 2017. 27, 28, 29, 34

[82] Consalta Cloud Enabler, "Top Cloud threats for 2017 and how vendors will respond with security," 2017. Last accessed, October 2018. Found Online at https://partner.microsoft.com/pl-pl/training/azuresalesstarprogram/top-cloud-threats-2017-and-how-vendors-will-respond-with-security. 27

[83] Qiao Yan & F. Richard Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *IEEE Communications Magazine, vol. 53, no. 4, pp. 52-59*, 2015. 28

[84] Mohamed Idhammad et al., "Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest," *Security and Communication Networks, vol. 2018*, 2018. 28, 38, 48, 50

[85] Radware, *DDoS Survival Handbook*. Published by Radware, Ltd, 2013. 28, 30, 31, 33, 34, 35, 36, 38, 41, 45, 53

[86] Dhruba Kumar Bhattacharyya & Jugal Kumar Kalita, *DDoS Attacks: Evolution, Detection, Prevention, Reaction, and Tolerance*. CRC Press. Taylor Francis Group, LLC, 2016. 28, 29, 30, 37, 38

[87] William Stallings, *Network Security Essentials: Applications and Standards*. Pearson Education, Inc., 6th edition ed., 2016. 29, 35, 36, 37

[88] Georgious Loukas & Gülay Öke, "Protecting against a Denial of Service Attack: A survey," *The Computer Journal. Vol. 53, no. 7, pp 1020–1037*, 2009. 29

[89] David Bisson, "The 5 Most Significant DDoS Attacks of 2016," 2016. Last accessed, October 2018. Found Online at https://www.tripwire.com/state-of-security/security-data-protection/cyber-security/5-significant-ddos-attacks-2016/. 29, 69

[90] Junhan Park et al., "Analysis of Slow Read DoS Attack and Countermeasures on Web servers," *International Journal of Cyber-Security and Digital Forensics, The Society of Digital Information and Wireless Communications. Vol. 4, pp. 339-353*, 2015. 29, 42, 45, 69

[91] Cisco Systems, Inc., "A Cisco Guide to Defending Against Distributed Denial of Service Attacks," *Cisco Security portal*. 29, 30, 31, 35, 36, 41, 47

[92] Hossein Bidgoli, *Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications*. Wiley Online Library, Printed: 2007, Online: 2012. 29, 30, 48

[93] Jelena Mirkovic & Peter Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms," 2004. 30, 31, 32, 35

[94] David Karig & Ruby Lee, "Remote Denial of Service Attacks and Countermeasures," *Princeton University Department of Electrical Engineering Technical Report CE-L2001-002*, 2001. 30, 31

[95] Arbor Networks, Inc, "What is DDoS," Last accessed, October 2018. https://www.netscout.com/what-is-ddos. 30, 35

[96] Stephen M. Specht & Ruby B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures," *International Workshop on Security in Parallel and Distributed Systems, p. 543-550*, 2004. 31, 32

[97] Mitko Bogdanoski et. al., "Analysis of the SYN Flood DoS Attack," *I.J. Computer Network and Information Security, 2013. Vol. 8, p. 1-11*, 2013. 33

[98] J. Postel, "Internet Control Message Protocol," 1981. Last accessed, October 2018. Found Online at https://tools.ietf.org/html/rfc792. 34

[99] Esraa Alomari et al., "Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art," *International Journal of Computer Applications Vol. 49, No.7, p.24-32*, 2012. 34

[100] R. Fielding & J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," 2014. Last accessed, October 2018. Found Online at https://tools.ietf.org/html/rfc7230. 34, 42

[101] R. Fielding & J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," 2014. Last accessed, October 2018. Found Online at https://tools.ietf.org/html/rfc7231. 34, 43, 44

[102] Saman Taghavi Zargar et al., "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communication surveys & Tutorials. Vol. 15, no. 4, pp. 2046 - 2069*, 2013. 34

[103] , "Alert (TA13-088A) DNS Amplification Attacks," Published: 2013. Revised: 2016. Last accessed, October 2018. Found Online at https://www.us-cert.gov/ncas/alerts/TA13-088A. 35

[104] Claude Fachkha et al., "Fingerprinting Internet DNS Amplification DDoS Activities," *The Sixth IFIP International Conference on New Technologies, Mobility and Security (NTMS 2014)*, 2014. 35

[105] David Holmes, "F5 DDoS Protection: Recommended Practices (Volume 1)," *White Paper*, 2013. 35, 41, 46, 47

[106] David Senecal, "Slow DoS on the Rise," 2013. Last accessed, October 2018. https://blogs.akamai.com/2013/09/slow-dos-on-the-rise.html. 35, 46

[107] Microsoft Corporation, "Security guidelines to detect and prevent DOS attacks targeting IIS/Azure Web Role (PAAS)," 2014. Last accessed, October 2018. Found Online at https://blogs.msdn.microsoft.com/friis/2014/12/30/security-guidelines-to-detect-and-prevent-dos-attacks-targeting-iisazure-web-role-paas/. 35, 40

[108] R. Shirey, "Internet Security Glossary, Version 2," 2007. Last accessed, October 2018. Found Online at https://tools.ietf.org/html/rfc4949. 35

[109] Chirag Sheth & Rajesh Thakker, "Performance Evaluation and Comparison of Network Firewalls under DDoS Attack," *International Journal of Computer Network and Information Security. Vol. 5, no. 12, pp. 60-67*, 2013. 36

[110] Diogo A. B. Fernandes et. al., "Security issues in cloud environments: a survey," *Springer-Verlag Berlin Heidelberg*, 2013. 36, 41

[111] Hung-Jen Liao et al., "Intrusion detection system: A Comprehensive Review," *journal of Network and Computer Applications, vol. 36, no. 1, pp. 16-24*, 2012. 36, 37

[112] Keke Gai et al., "Intrusion Detection Techniques for Mobile Cloud Computing in Heterogeneous 5G," *Security and Communication Networks, vol. 9, no. 16, pp. 3049-3058*, 2015. 37

[113] Ashley Chonka et al., "Cloud Security Defence to Protect Cloud Computing against HTTP-DoS and XML-DoS Attacks," *Journal of Network and Computer Applications, vol. 34, no. 4, pp. 1097-1107*, 2011. 38

[114] Lawrence Koved & Lin Luo, "Interactive management of web application firewall rules," 2016. Last accessed, October 2018. Found Online at https://www.google.com/patents/US9473457. 39

[115] Jinjin Liang et al., "When HTTPS Meets CDN: A Case of Authentication in Delegated Service," *IEEE Symposium on Security and Privacy*, 2014. 39, 40

[116] Igor Ljunbucic, *Apache Web Server Complete Guide*. Software and Security, Published Online on Dedoimedo Website, 2011. 39

[117] Dan Breslaw, "Configuring mod_evasive to Protect Apache Servers," 2016. Last accessed October, 2018. Found online at https://www.incapsula.com/blog/configuring-mod_evasive-to-protect-your-apache-server.html. 39

[118] Pascal Buchbinder, "mod_qos," 2007. Last accessed, July 2017. Found Online on http://mod-qos.sourceforge.net/. 40

[119] The Apache Software Foundation, "Apache mod_reqtimeout," Last accessed, October 2018. Found online at https://httpd.apache.org/docs/trunk/mod/mod_reqtimeout.html. 40

[120] Changwang Zhang et al., "Flow level detection and filtering of low-rate DDoS," *Computer Networks: The International Journal of Computer and Telecommunications Networking. Vol. 56, no. 15, pp. 3417-3431*, 2012. 42, 48, 49

[121] Ian Muscat, "How To Mitigate Slow HTTP DoS Attacks in Apache HTTP Server," 2013. Last accessed, October 2018. Found Online at https://www.acunetix.com/blog/articles/slow-http-dos-attacks-mitigate-apache-http-server/. 42, 43, 48

[122] Robert "RSnake" Hansen & John Kinsella, "Slowloris HTTP DoS," 2009. October 2018. Found Online at https://web.archive.org/web/20150315054838/http://ha.ckers.org/slowloris/. 42, 43, 44, 60

[123] Sergey Shekyan, "Tweaking to get away from SlowDOS," 2012. Last accessed, October 2018. Found Online at https://www.owasp.org/images/a/a6/Owasp_KS_slowDoS.pdf. 42, 46

[124] Zenghui Liu & Liguo Guan, "Attack simulation and signature extraction of low-rate DoS," *Third International Symposium on Intelligent Information Technology and Security Informatics. pp. 544-548*, 2010. 42, 49

[125] Robert ”RSnake” Hansen, “Slowloris,” 2012. Last accessed, October 2018. Found Online at https://samsclass.info/seminars/slowloris.pdf. 42, 44, 48, 60

[126] Wong Onn Chee & Tom Brennan, “H.....t.....t....p....p....o....s....t,” 2010. Last accessed, October 2018. Found Online at https://www.owasp.org/images/4/43/Layer_7_DDOS.pdf. 42, 44

[127] Maryam M. Najafabadi et al., “RUDY Attack: Detection at the Network Level and Its Important Features,” *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference. pp. 282-287*, 2016. 45, 48

[128] Sergey Shekyan, “slowhttptest,” 2011-2016. Last accessed, October 2018. Found Online at https://github.com/shekyan/slowhttptest/wiki. 45, 55, 60

[129] Evan Damon et. al., “Hands-On Denial of Service Lab Exercises Using Slowloris and RUDY,” *SInformation Security Curriculum Development Conference*, 2012. 45, 48

[130] Sergey Shekyan, “Identifying Slow HTTP Attack Vulnerabilities on Web Applications,” 2011. Last accessed, October 2018. Found Online on https://blog.qualys.com/securitylabs/2011/07/07/identifying-slow-http-attack-vulnerabilities-on-web-applications. 47

[131] Monowar H. Bhuyan et al., “Information Metrics for Low-rate DDoS Attack Detection: A Comparative Evaluation,” *Seventh International Conference on Contemporary Computing (IC3)*, 2014. 48

[132] Monowar H. Bhuyan et al., “An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection,” *Pattern Recognition Letters. Vol. 51, pp. 1-7*, 2014. 49

[133] M. Aiello et al., “An On-Line Intrusion Detection Approach to Identify Low-Rate DoS Attacks,” *International Carnahan Conference on Security Technology (ICCST)*, 2014. 50

[134] Yuri G. Dantas et al., “A Selective Defense for Application Layer DDoS Attacks,” *IEEE Joint Intelligence and Security Informatics Conference*, 2014. 50

[135] Markus Ring et al., “Flow-based benchmark data sets for intrusion detection,” *in Proceedings of the in Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS). pp. 361-369*, 2017. 51

[136] opennebula.org., “OpenNebula: VirtualBox SandBox,” Last accessed, October 2018. Found Online at https://opennebula.org/tryout/sandboxvirtualbox/. 54

[137] RedHat. Found Online at https://www.okd.io/. 55

[138] Gerald Combs et al., “Tshark,” 1998, Last accessed, October 2018. Found Online at https://www.wireshark.org/docs/man-pages/tshark.html. 55

[139] Gerald Combs et al., “The Wireshark Analyzer,” 1998. Last accessed, October 2018. Found Online at https://www.wireshark.org/docs/man-pages/tshark.html. 57

Appendix A

Appendix A

A.1 Codes

A.1.1 Structure to Store Values from Each Packet

```
typedef  struct  s_packet{
    char *sourceIp;
    char *destIp;
    int windowSize;
    int contentLength;
    int frameLength;
    int responseCode;
    int startPosition;
    int endPosition;
    int count;
}PacketInfo;
```

A.1.2 Store Values Inside Structure

```
    int tokenCounter,read;
    while (1) {
        tokenCounter = 0;
        // Returns first token
        v = realloc(v,(*n+1) * sizeof(PacketInfo));
        read = fscanf(fp,"%s\n",buffer);
        if(read < 1) break;

        char *token = strtok(buffer, ";");
        int flag = 0;
        // Keep printing tokens while one of the
        // delimiters present in str[].
        while (token != NULL) {

            //printf("%s\t\t", token);
            token = strtok(NULL, ";");
            if(token == NULL){
                (*n)++;
                break;
            }
            switch (tokenCounter){
                case 0:
```

```
                v[*n].sourceIp = malloc(strlen(token) * sizeof(char));
                strncpy(v[*n].sourceIp,token,strlen(token));
                tokenCounter++;
                if(v[*n].sourceIp == NULL) {
                    flag = 1;
                }
                break;
            case 1:
                v[*n].destIp = malloc(strlen(token) * sizeof(char));
                strncpy(v[*n].destIp,token,strlen(token));
                tokenCounter++;
                break;
            case 2:
                v[*n].frameLength = atoi(token);
                tokenCounter++;
                break;
            case 3:
                v[*n].windowSize = atoi(token);
                tokenCounter++;
                break;
            case 4:
                v[*n].responseCode = atoi(token);
                tokenCounter++;
                break;
            case 5:
                v[*n].contentLength = atoi(token);
                tokenCounter++;
                break;
            default:
                printf("Content not saved\n");
                break;
        }
    }
```